D-RisQ

SOFTWARE SYSTEMS

Changing the way the world does software

D-RisQ
SOFTWARE SYSTEMS

ED-12C Compliant Autonomous Decision Making for UAVs

Nick Tudor

njt@drisq.com

D-RisQ SOFTWARE SYSTEMS

Overview

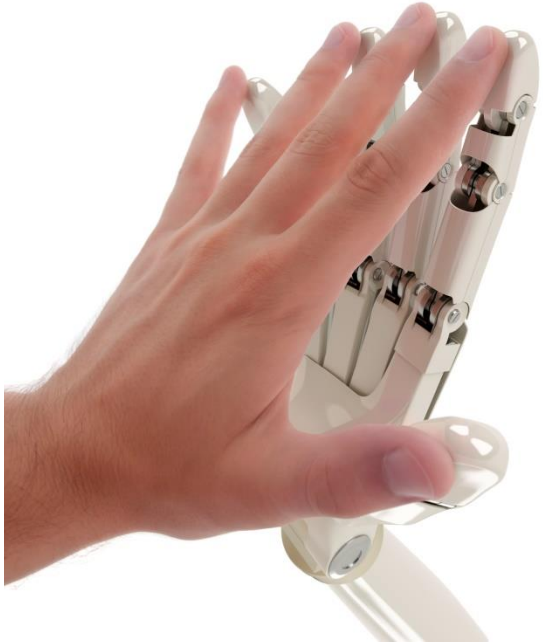
- Motivation for the project
- Focus on costs and certification (ie safety)
- Overview of the software development and verification
- Results
- Future work
- Wrap up

D-RisQ
SOFTWARE SYSTEMS

Difference?

‘Autonomous’

People



Machines

Attention paid in real time …..&….. How we communicate

MOTIVATION

D-RisQ SOFTWARE SYSTEMS

Beyond Visual Line of Sight (BVLOS)

- Aircraft in line of sight are under control of the user
  - User expected to react to issues and is responsible
- BVLOS the aircraft has to be able to react to issues without intervention by the user
  - Implies a really complex piece of software

Also implies an expensive piece of software!

Behaviour must be as another air user would expect it to react

FOCUS ON COSTS AND CERTIFICATION

D-RisQ
SOFTWARE SYSTEMS

Making/Accessing a Market

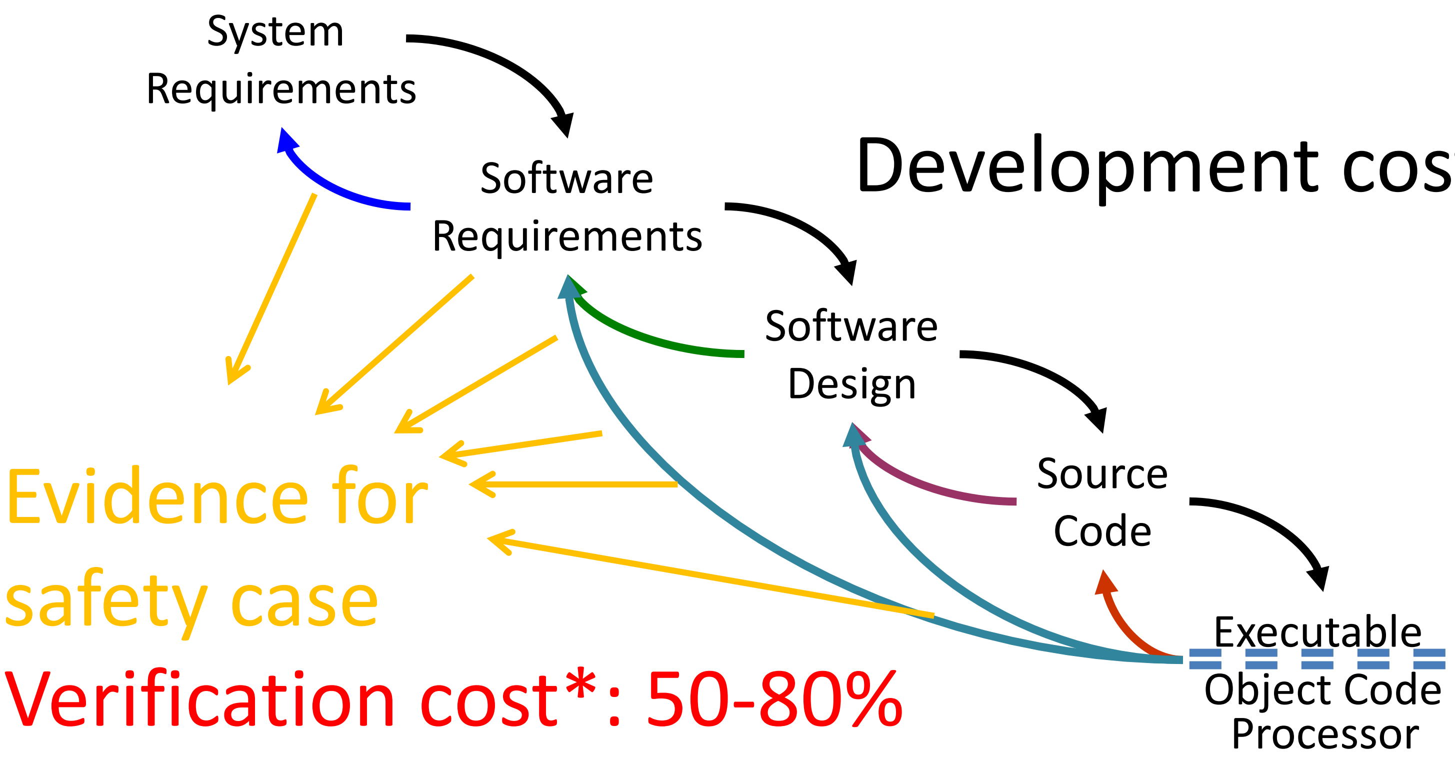
How to make kit affordable to the market?

How to make sure the kit is acceptable to the market?

Does ‘acceptable’ mean functionality as well as safety?

Systems, Software and Certification

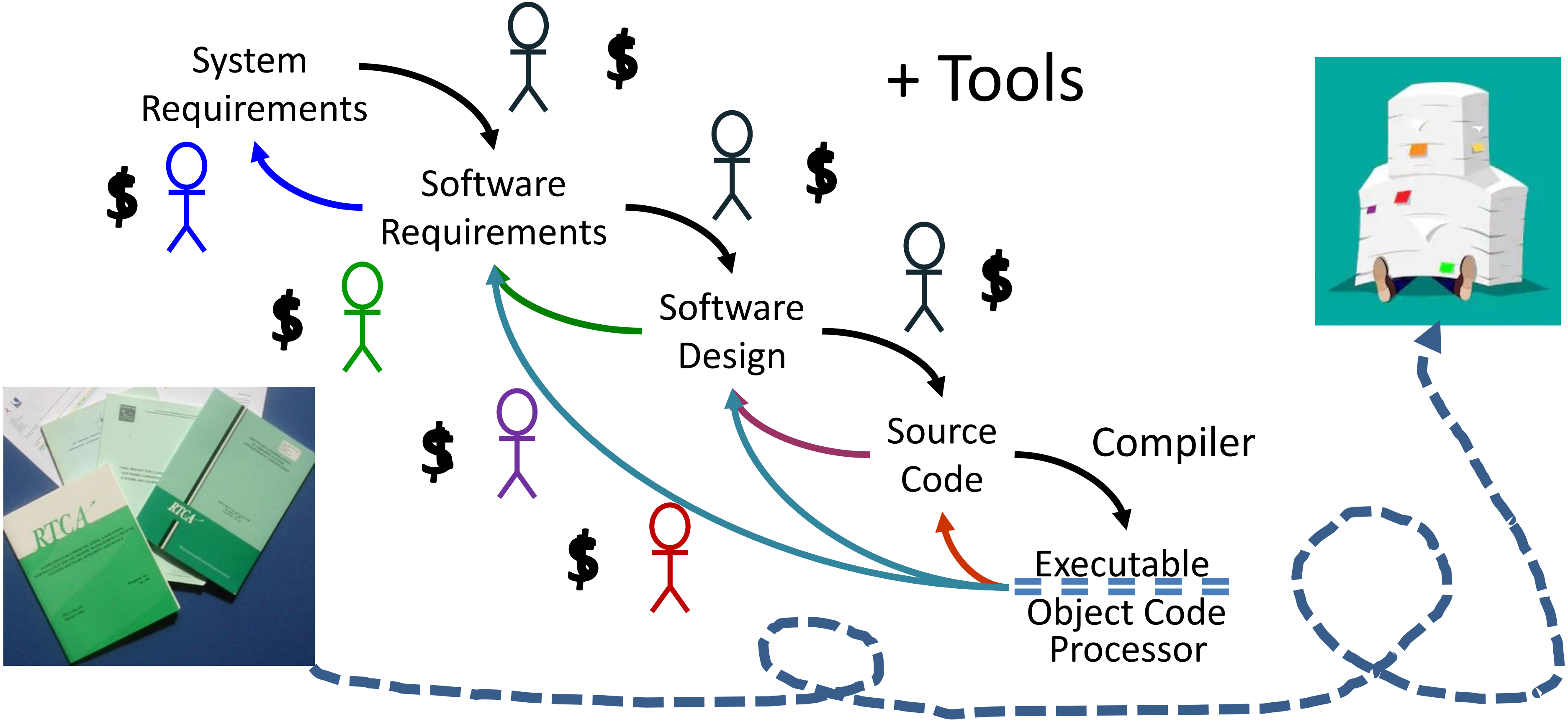
Development cost*: 20-50%



Verification cost*: 50-80%

* % of total: Variation caused mainly by integrity level

Systems, Software and Certification



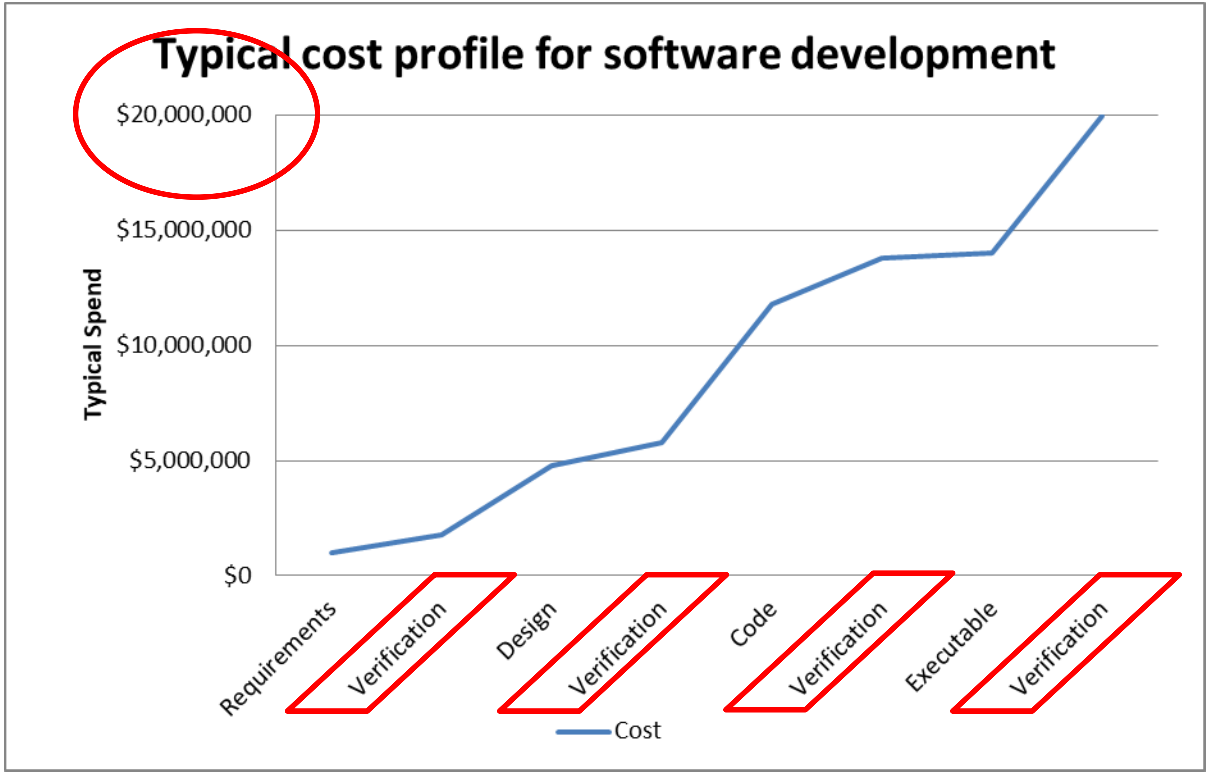
D-RisQ SOFTWARE SYSTEMS

Forum for Aeronautical Software

- Asked by EUROCAE/RTCA to provide a white paper on use of ED-12C for UAS
  - Published Feb 2019
- Engaged with participants in JARUS WG
  - Confused about DAL vs Software Level
  - Also confusion over how to use these in design
  - Weight issues and use cases not well thought out in SORA
- Enabling BVLOS decision making system will be safety critical and implies DAL A system and Level A software

Typical Project Costs (Level A)

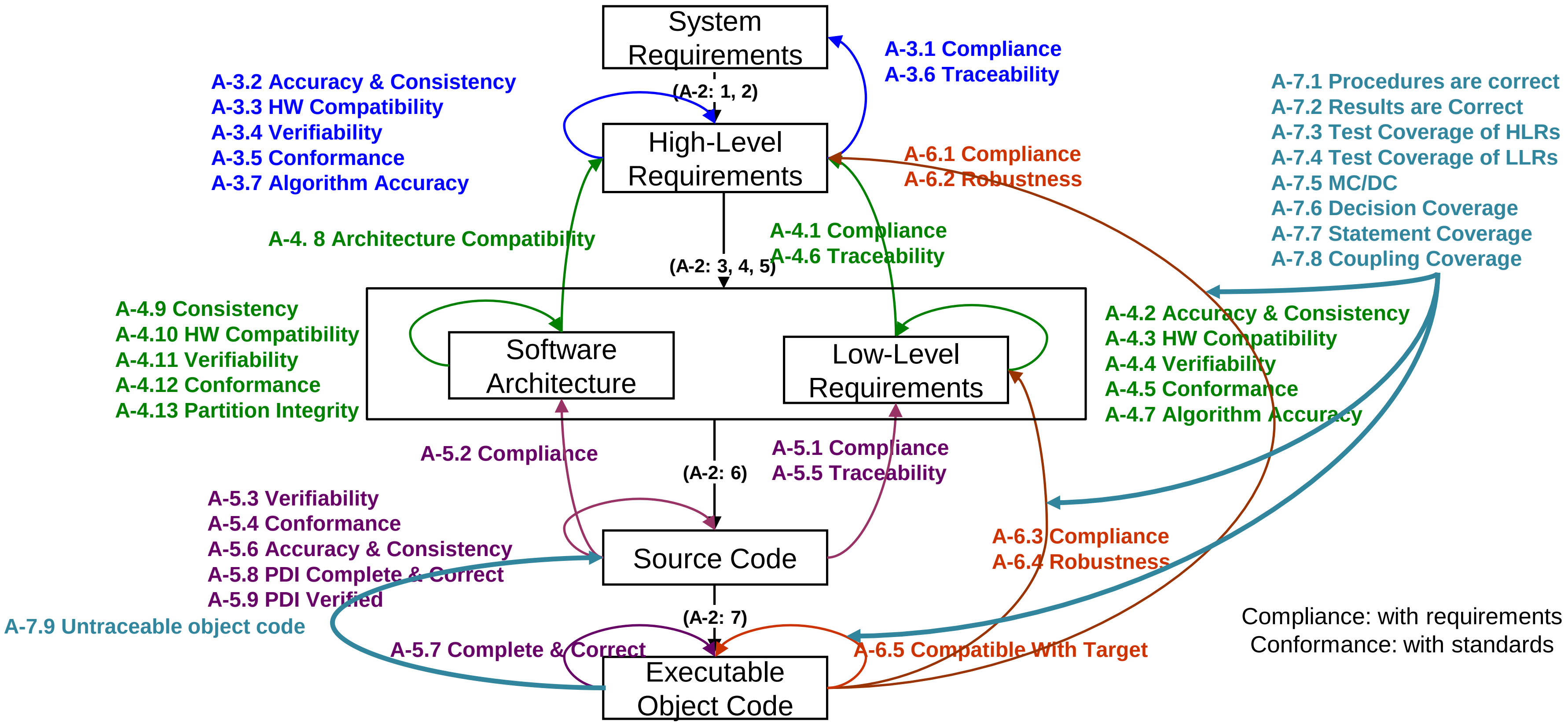
Barrier to market entry



Cost profile independently validated by York Metrics

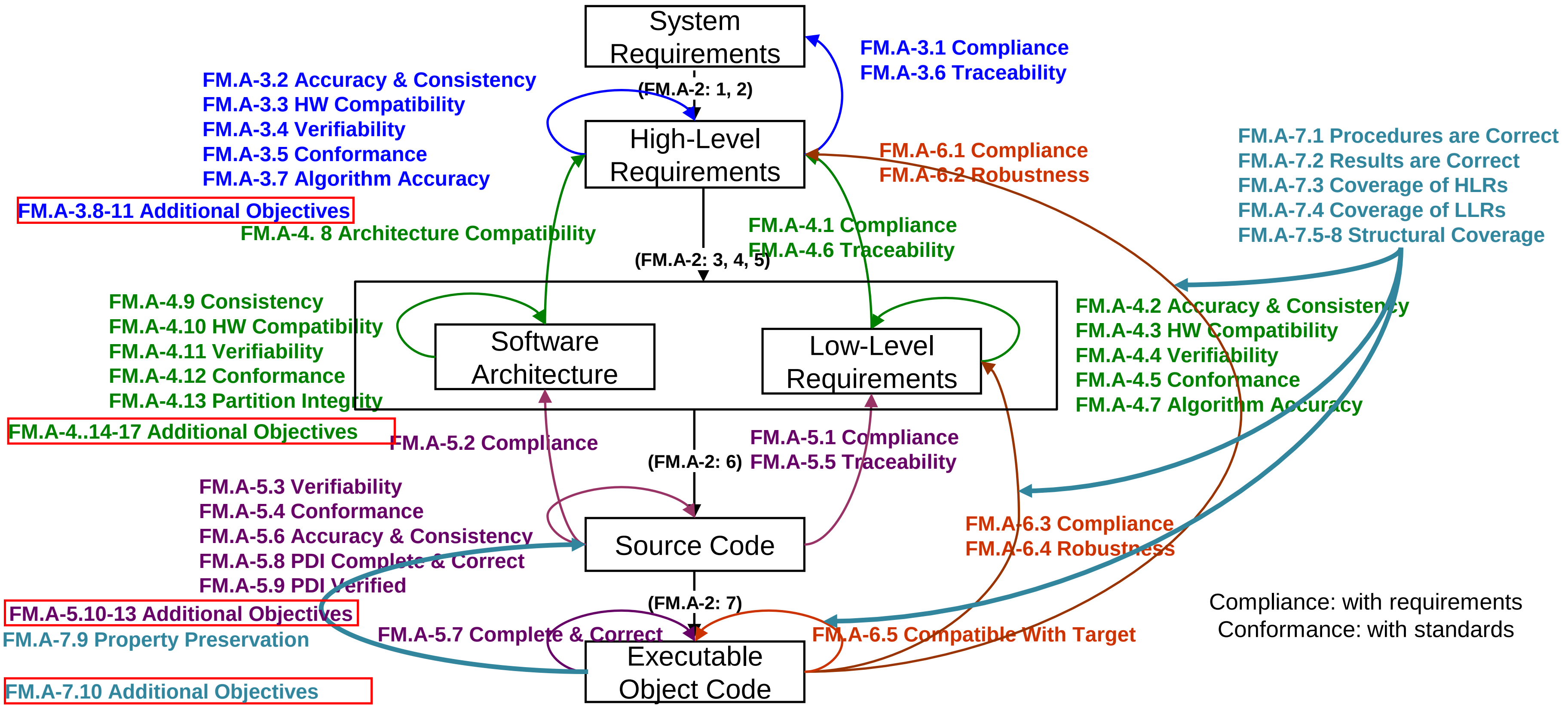
RTCA DO-178C/EUROCAE ED-12C & ASSOCIATED DOCUMENTS

Verification Objectives – DO-178C

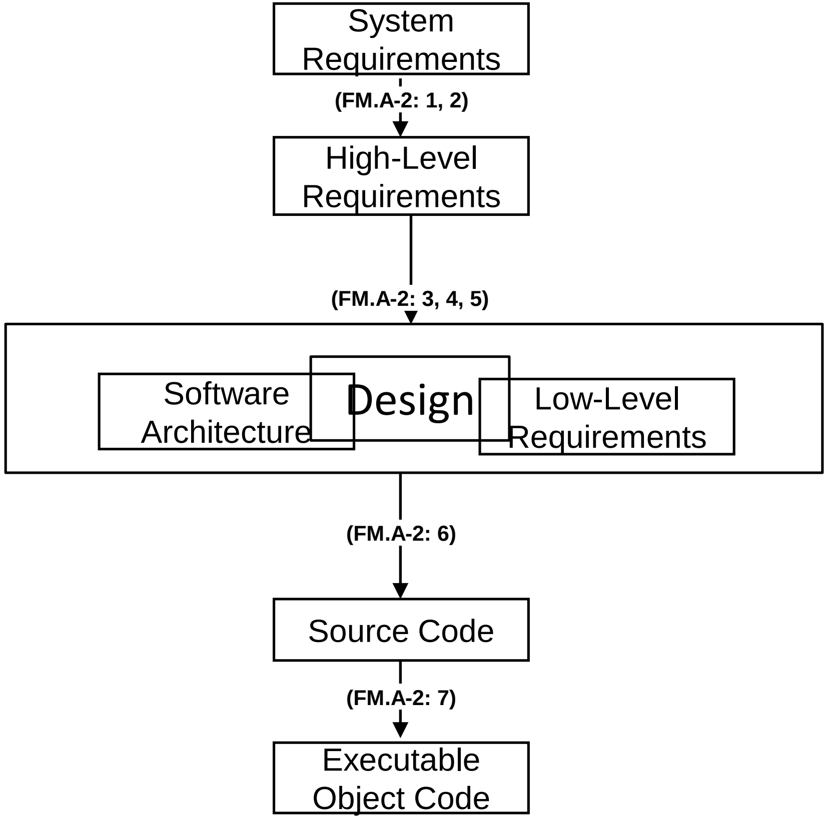


Compliance: with requirements
Conformance: with standards

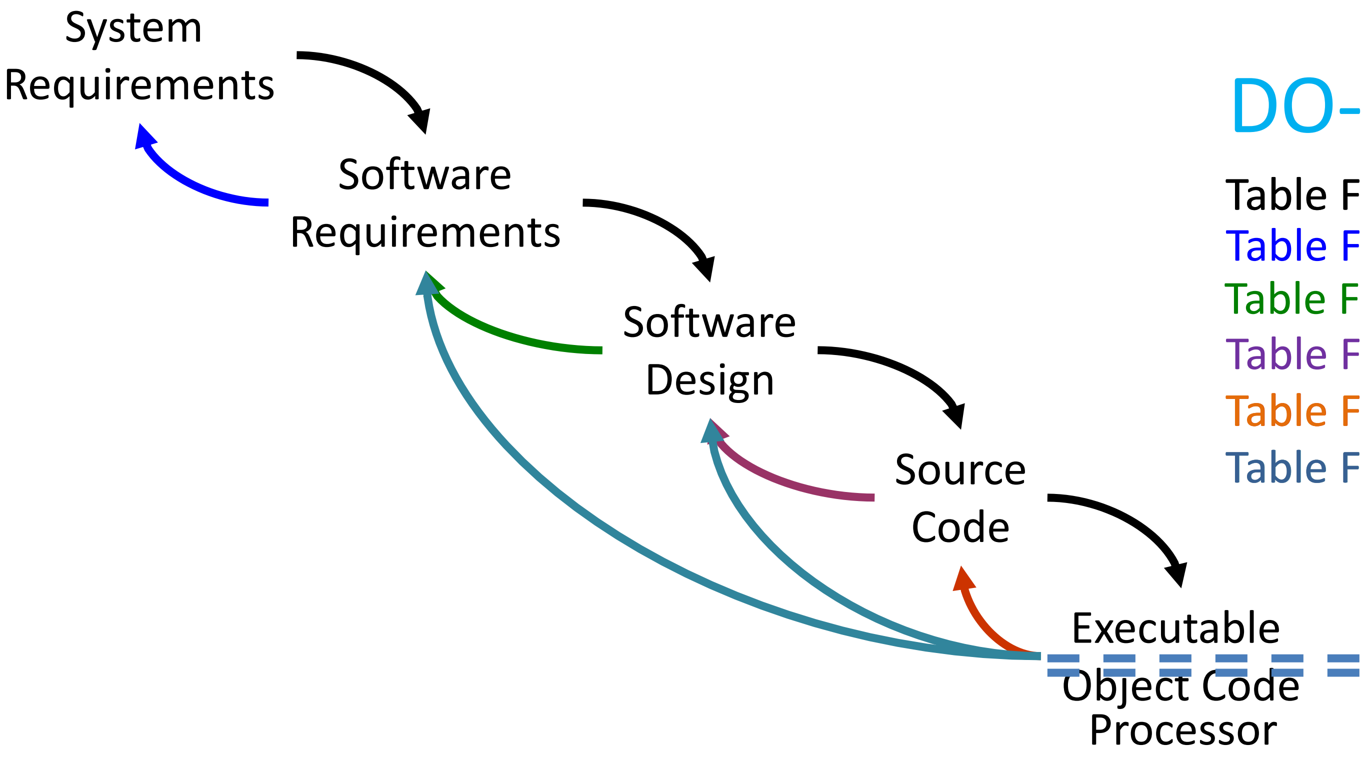
Verification Objectives – DO-333



Verification Objectives – DO-333



Systems, Software and Certification



DO-333

Table FM.A-2 Objectives
Table FM.A-3 Objectives
Table FM.A-4 Objectives
Table FM.A-5 Objectives
Table FM.A-6 Objectives
Table FM.A-7 Objectives

DO-178C – Section 4 Planning

- Section 4.4: Software Life Cycle Environment Planning
  - The goal of error prevention methods is to avoid errors during the software development processes that might contribute to a failure condition. The basic principle is to choose requirements development and design methods, tools, and programming languages that limit the opportunity for introducing errors, and verification methods that ensure that errors introduced are detected.
- Section 4.5.c: Software Development Standards
  - The software development standards should disallow the use of constructs or methods that produce outputs that cannot be verified or that are not compatible with safety-related requirements.

OVERVIEW OF THE SOFTWARE DEVELOPMENT AND VERIFICATION

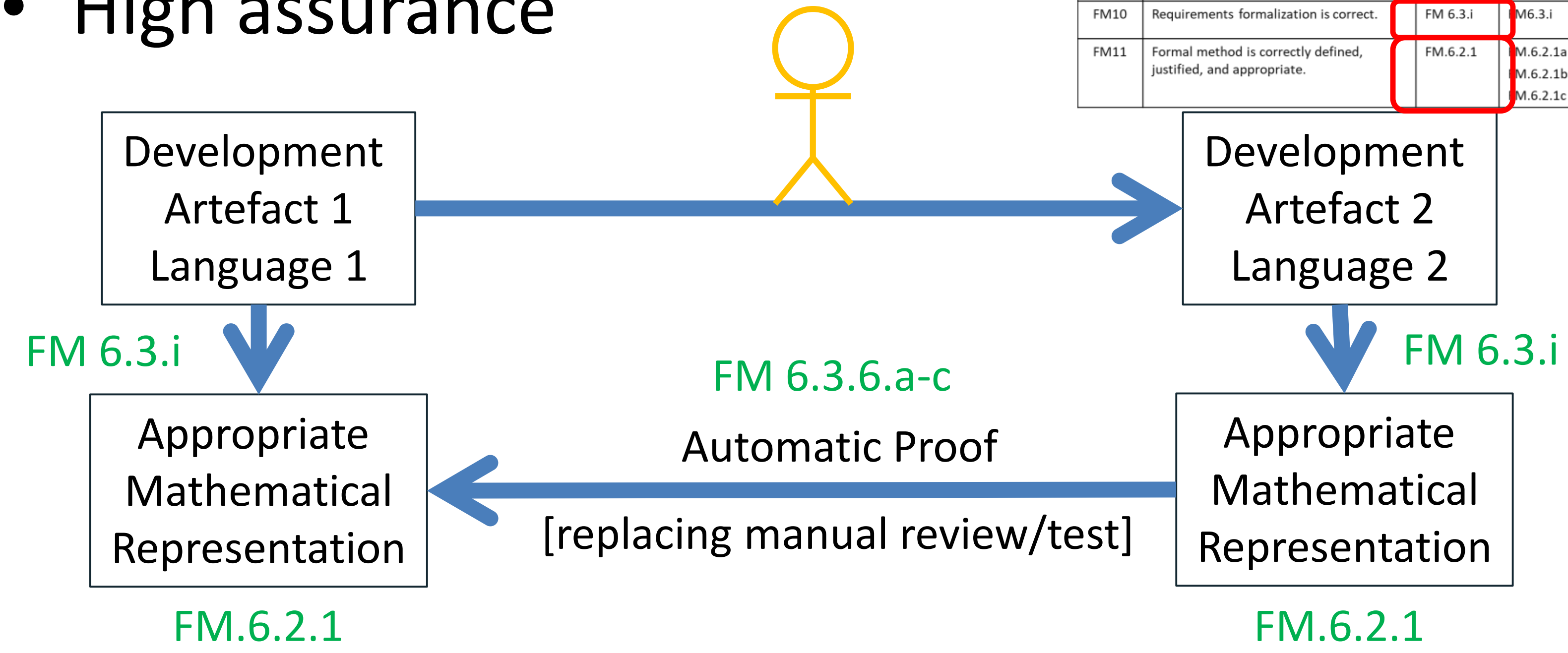
D-RisQ Product Principles

- Removal of opportunity for error introduction (Section 4.4)
- Enforce various standards to enable verification (Section 4.5.c)
- Use commercially available technology familiar in the market, then independently apply rigour
  - Enable adaptation of existing processes
  - Little or no re-training
- Use mathematical techniques to replace [statistics based] testing: enable proof
  - As far as possible, hide all the maths
- Provide evidence to support system certification
  - Includes IEC 61508, IEC 60880, EN50128/9, DO-178C, ISO26262…
- cf Laptop; you don’t need to know how a laptop works to be able to use it
- Aim is to develop licensable technology

Verification Approach

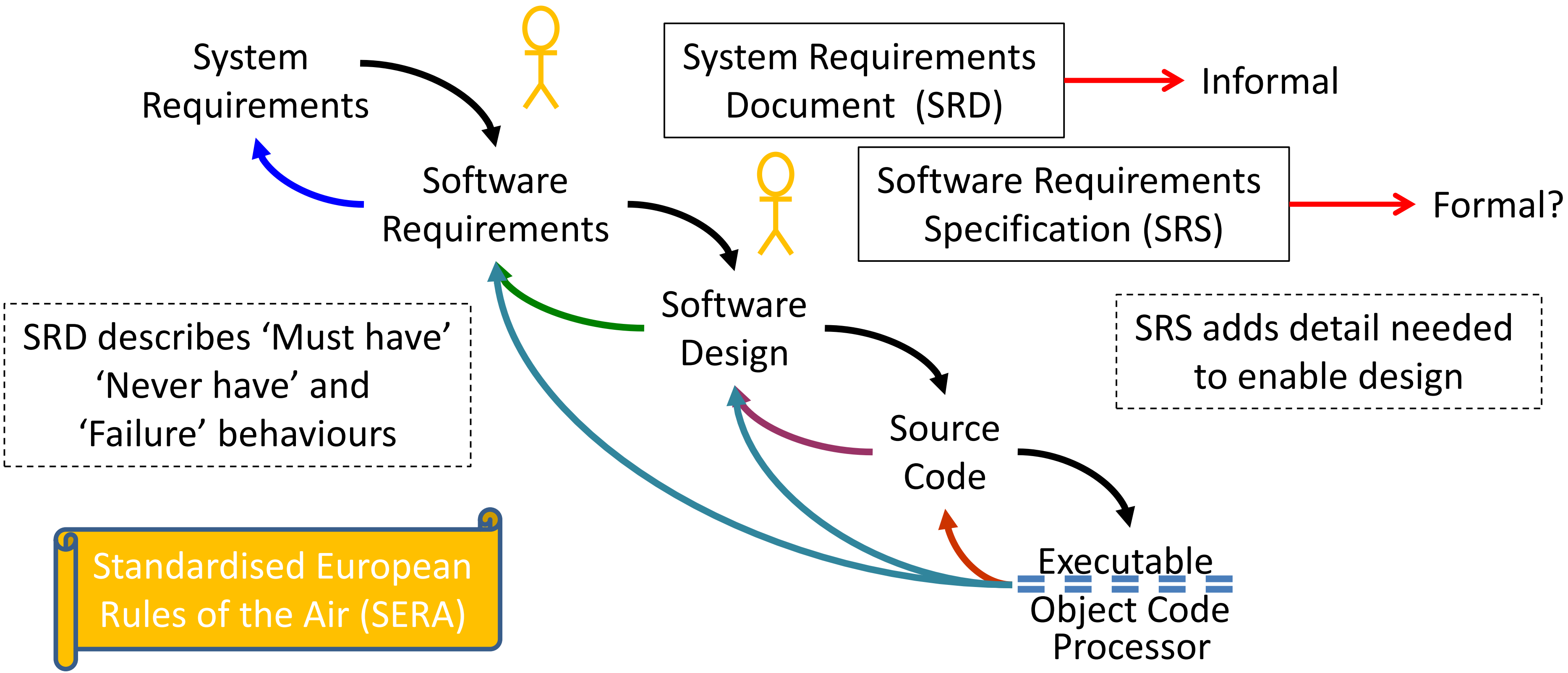
- Independence and automation
- High assurance

	Objective		Activity	Claim
	Description	Ref	Ref	
FM8	Formal analysis cases and procedures are correct.	FM 6.3.6.a FM 6.3.6.b	FM6.3.6	Tool Qualification and user procedures
FM9	Formal analysis results are correct and discrepancies explained.	FM 6.3.6.c	FM 6.3.6	Tool Qualification and user procedures
FM10	Requirements formalization is correct.	FM 6.3.i	FM6.3.i	Tool Qualification material supplied by D-RisQ
FM11	Formal method is correctly defined, justified, and appropriate.	FM.6.2.1	FM.6.2.1a FM.6.2.1b FM.6.2.1c	Tool Qualification material supplied by D-RisQ



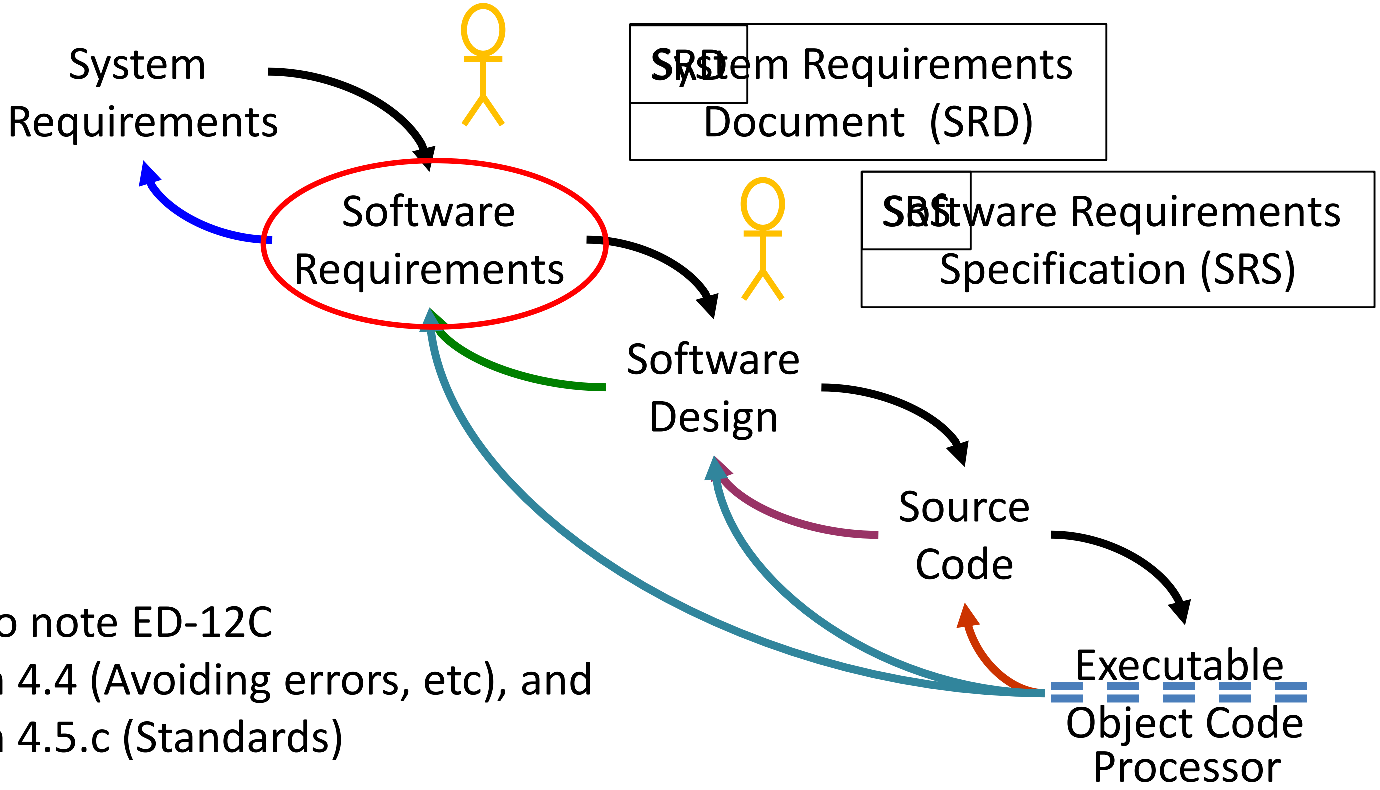
D-RisQ SOFTWARE SYSTEMS

Describing UAV Behaviour



D-RisQ
SOFTWARE SYSTEMS

Describing UAV Behaviour



Need to note ED-12C
Section 4.4 (Avoiding errors, etc), and
Section 4.5.c (Standards)

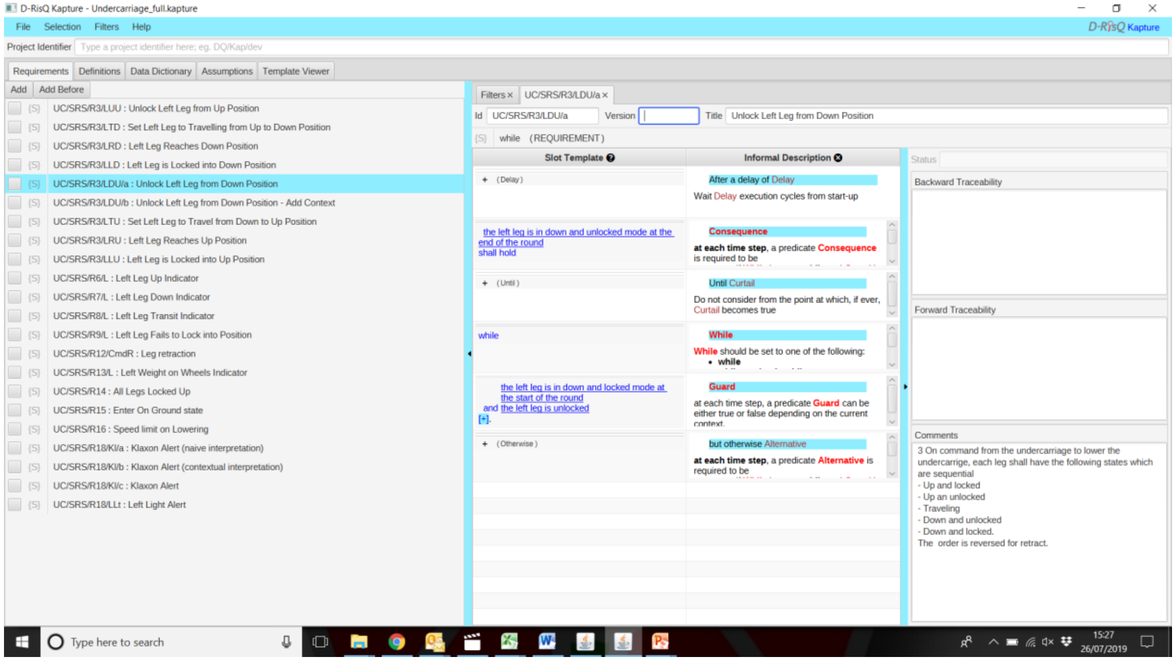
REQUIREMENTS - KAPTURE

D-RisQ
SOFTWARE SYSTEMS

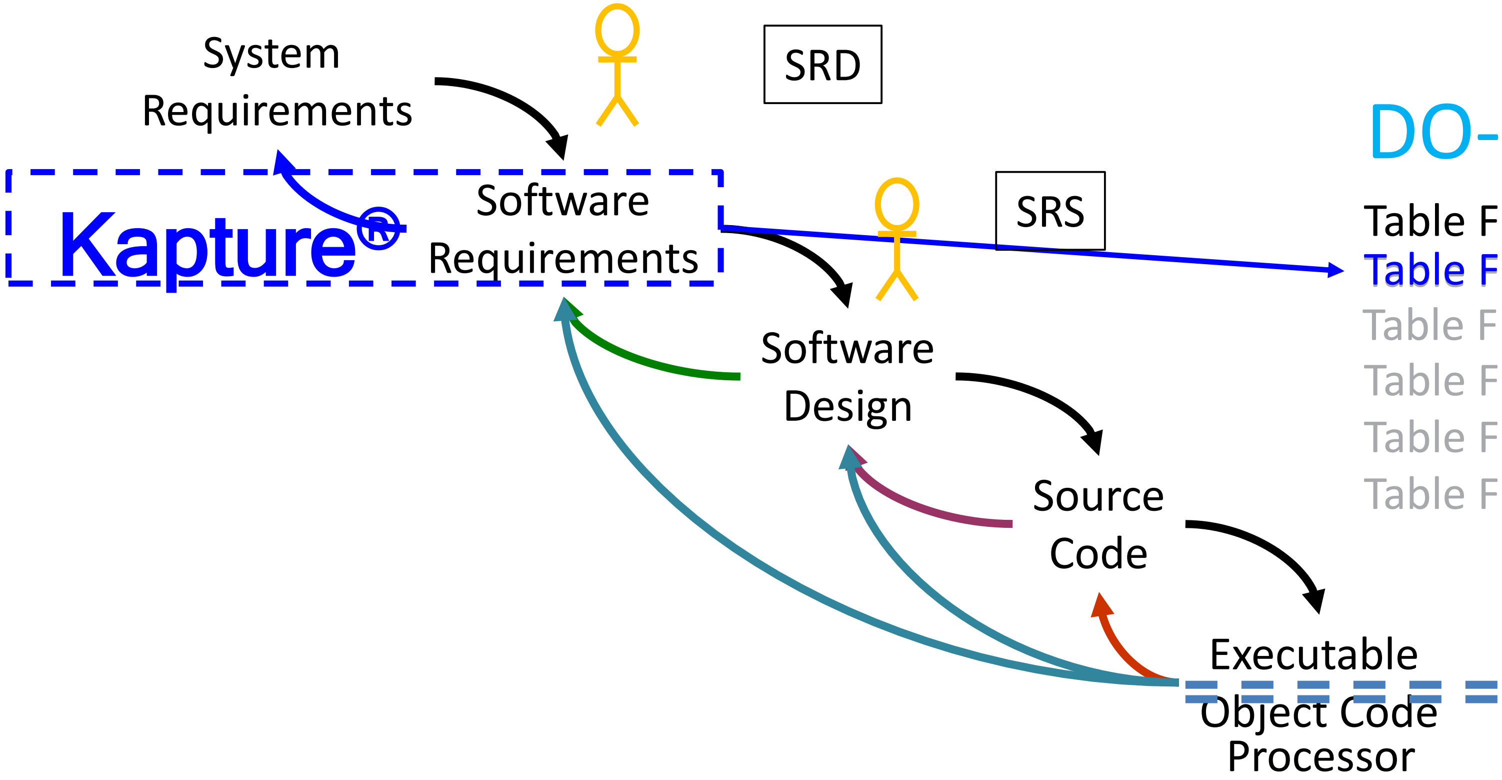
Kapture

Features:

- 6 requirement templates with various options (all 'verifiable')
- Separate data dictionary
- Definitions and Assumptions
- Offers drop down easily fill menu for text
- Help easily visible
- Export to various formats
- Various filters
- Expansion for System requirements link
- Assures 'healthiness' of requirements



Describing Behaviour



DO-333

Table FM.A-2 Objectives
Table FM.A-3 Objectives
Table FM.A-4 Objectives
Table FM.A-5 Objectives
Table FM.A-6 Objectives
Table FM.A-7 Objectives

D-RisQ SOFTWARE SYSTEMS

Table FM.A-3 – Verification of Output Design Process

	Objective		Activity	Claim
	Description	**Ref**	**Ref**	
1	High-level requirements comply with system requirements.	FM.6.3.a FM.6.3.1.a	FM.6.3.1	Manual review needed until Kapture for System Requirements
2	High-level requirements are accurate and consistent.	FM.6.3.b FM.6.3.c FM.6.3.1.b	FM.6.3.1	Basic functionality of Kapture supports accuracy claim; extra functionality gives consistency and unambiguity.
3	High-level requirements are compatible with target computer.	FM.6.3.d FM.6.3.1.c	FM.6.3.1	Kapture does not support this aspect: manual review
4	High-level requirements are verifiable.	FM.6.3.e FM.6.3.1.d	FM.6.3.1	Kapture requirements are verifiable due to the provision of semantics.
5	High-level requirements conform to standards.	FM.6.3.f FM.6.3.1.e	FM.6.3.1	Kapture encapsulates a requirements standard
6	High-level requirements are traceable to system requirements.	FM.6.3.g FM.6.3.1.f	FM.6.3.1	Manual review of manually entered data until Kapture for System Requirements
7	Algorithms are accurate.	FM.6.3.h FM.6.3.1.g	FM.6.3.1	Algorithm accuracy can be partially shown through the use of Kapture

Fully met | Partially met | Not met

D-RisQ SOFTWARE SYSTEMS

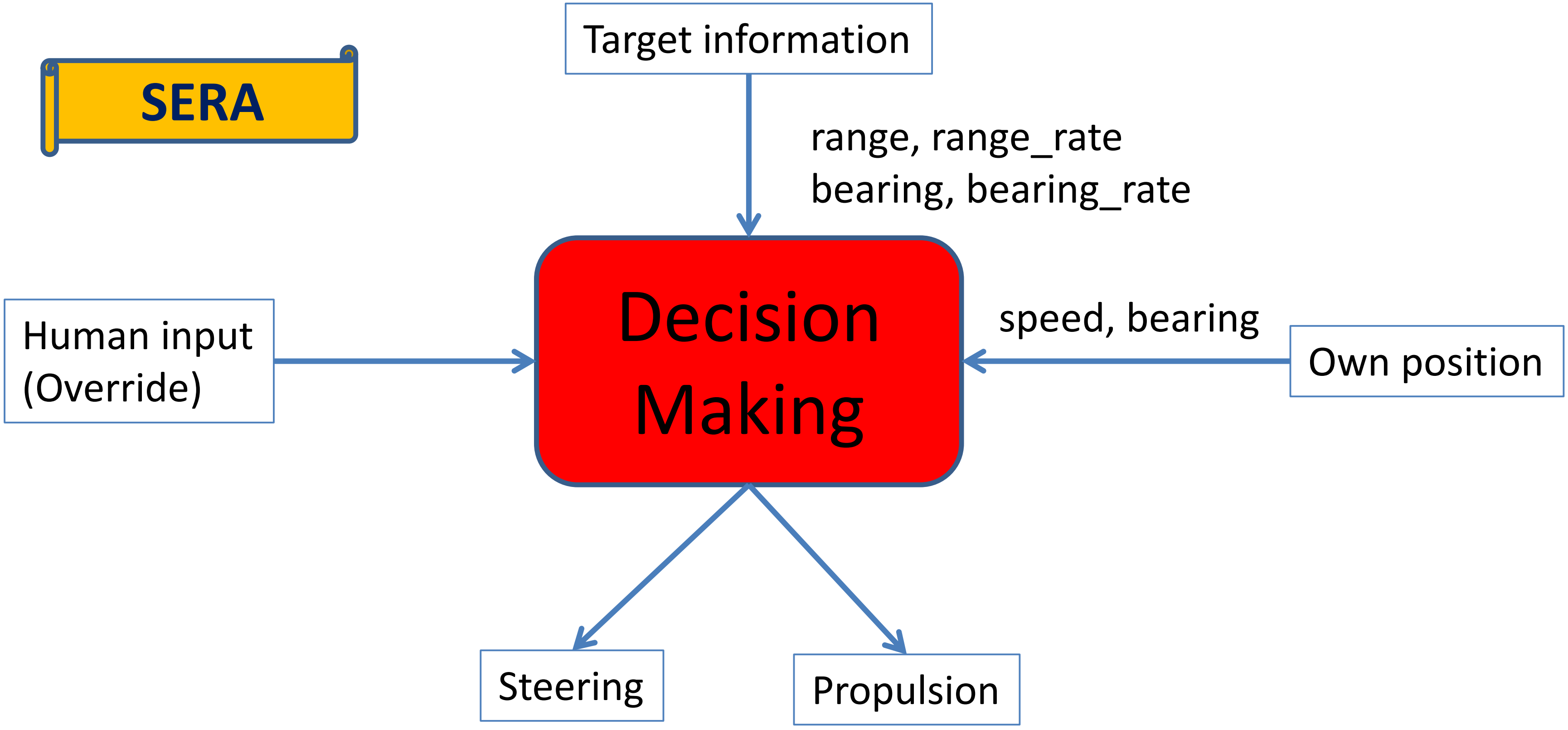
Software High Level Requirements

- These were developed in Kapture and formed the Software Requirements Specification (SRS)
  - Formal semantics given to English constructs
  - Validated behaviour
- Described the behaviour required in order to comply with SERA
  - Drop 1 basic functionality
  - Drop 2 gave extended behavioural capability; behaves as though ‘manned’
- Credit for certification can be taken or/and reviews done additionally
  - NB Has to be some review between SRS and SRD

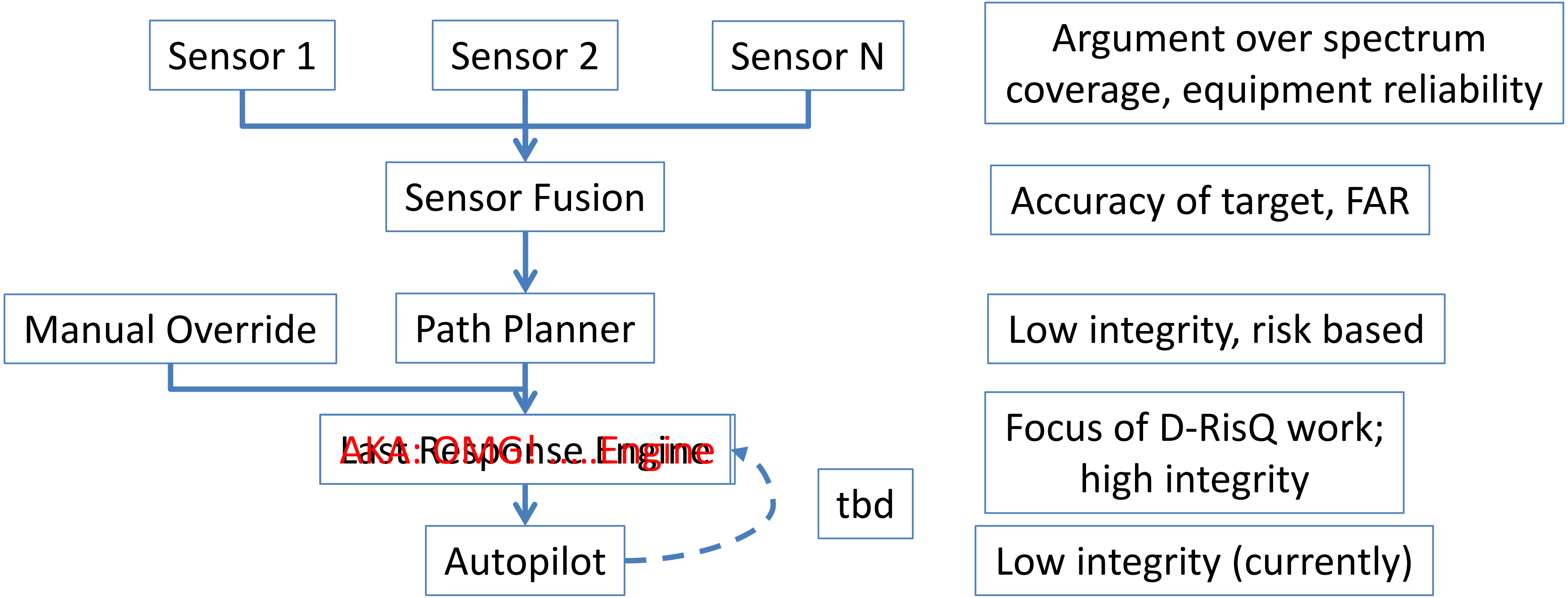
Off to → Software Design

DESIGN & VERIFICATION - MODELWORKS

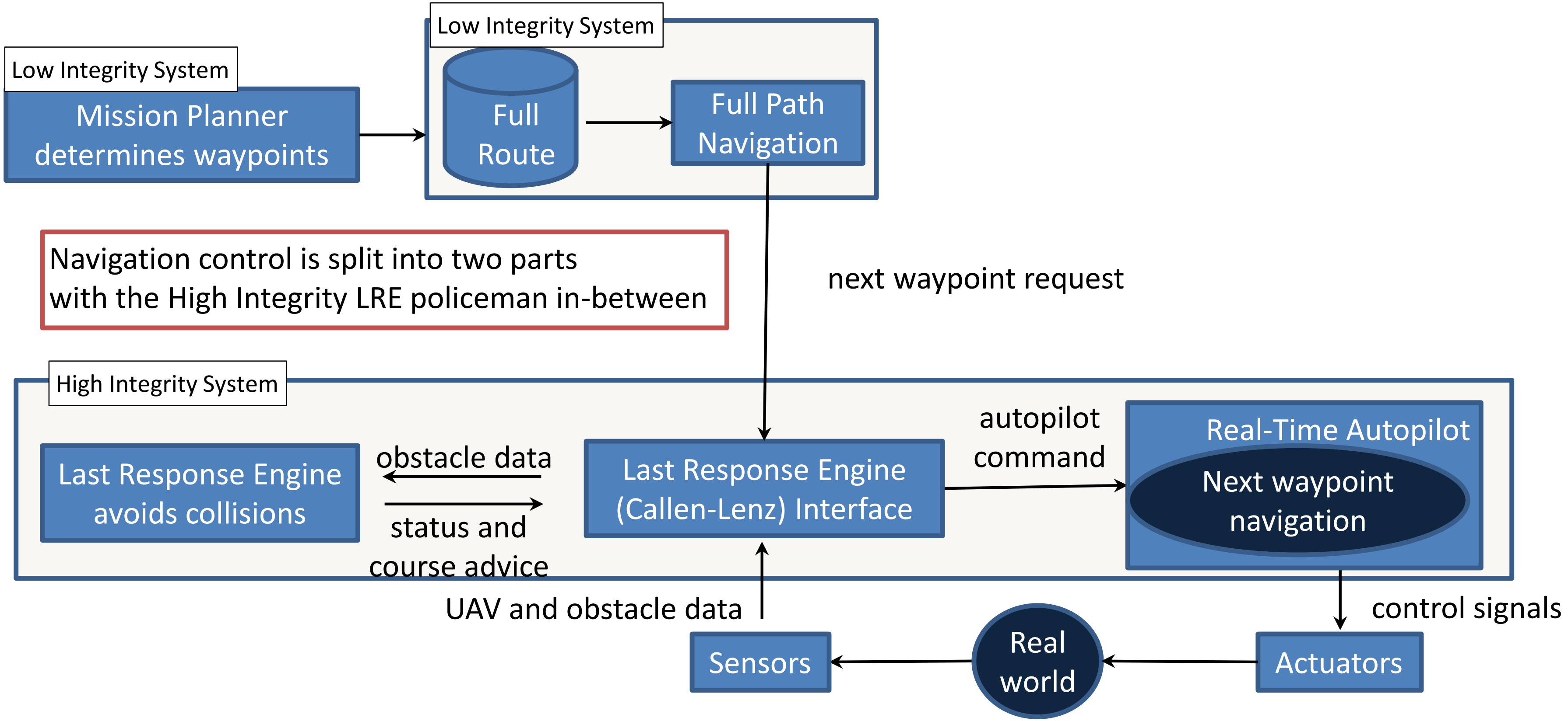
Decision Making System



A General Architecture

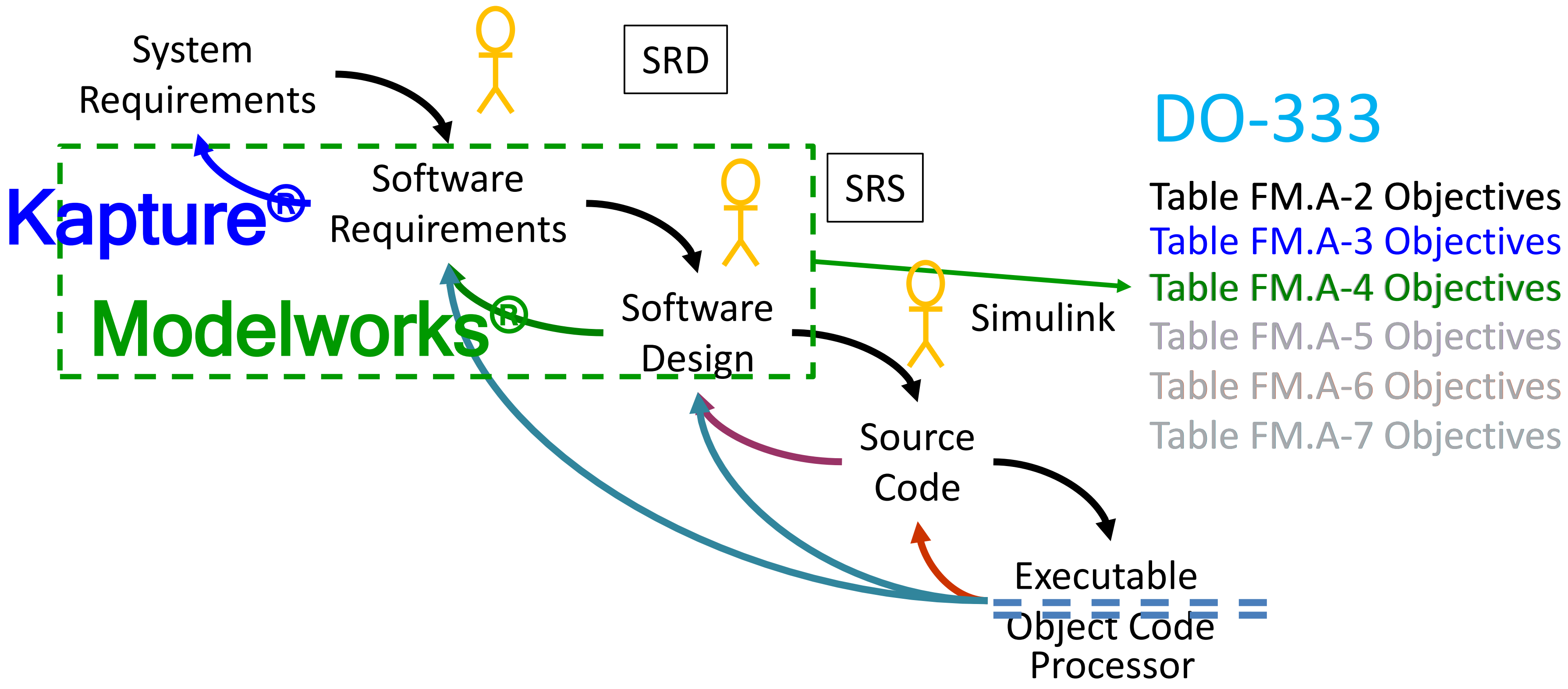


System Architecture



Navigation control is split into two parts
with the High Integrity LRE policeman in-between

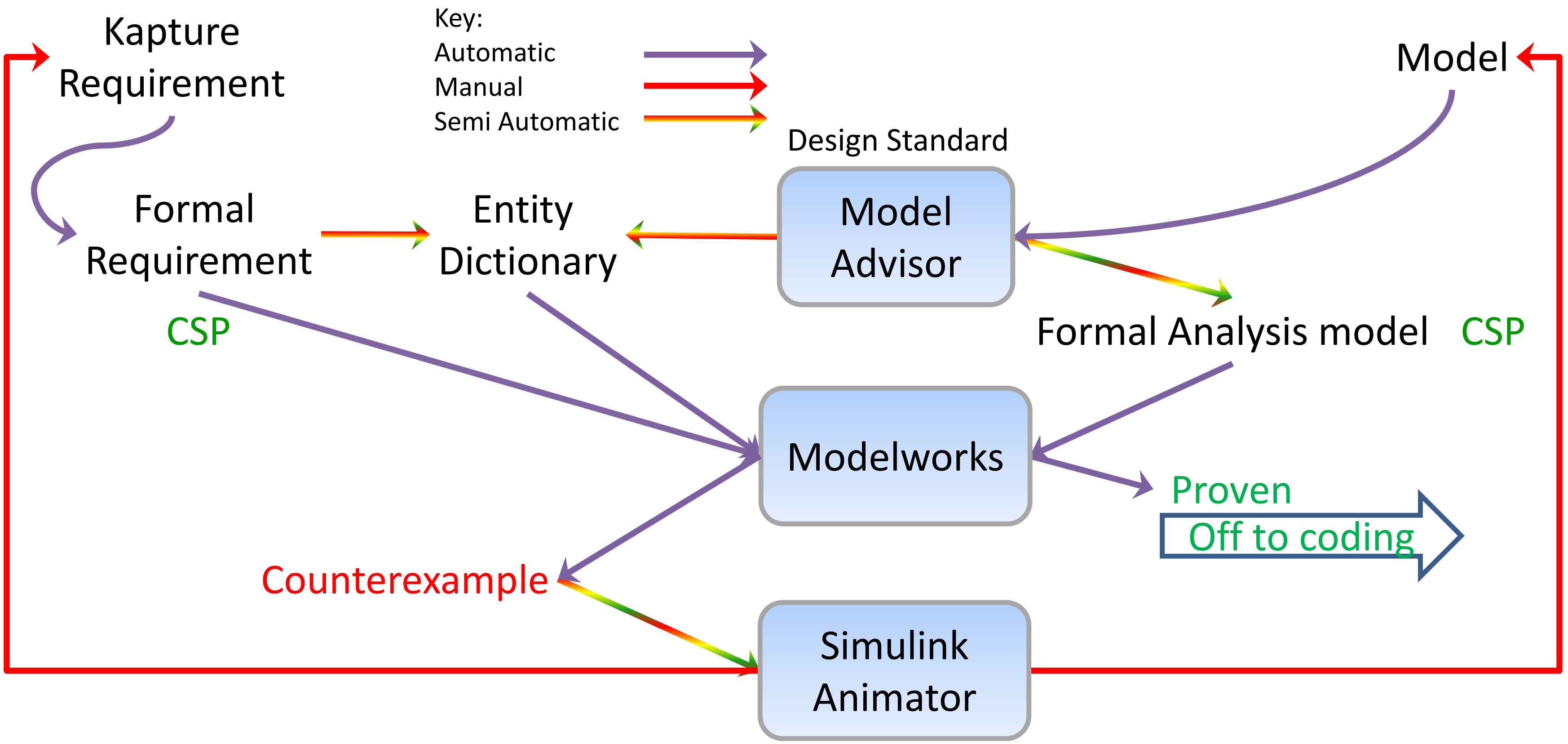
Systems, Software and Certification



Low Level Requirements/Architecture

- The design in Simulink was verified wherever possible using Modelworks
  - Some things not verifiable formally
  - "The software shall be developed to ED-12C Level A" is not formally verifiable
- In order to do this, both requirements and Simulink have semantics expressed in CSP

Modelworks Process



Modelworks & Table FM.A-4 - LLR

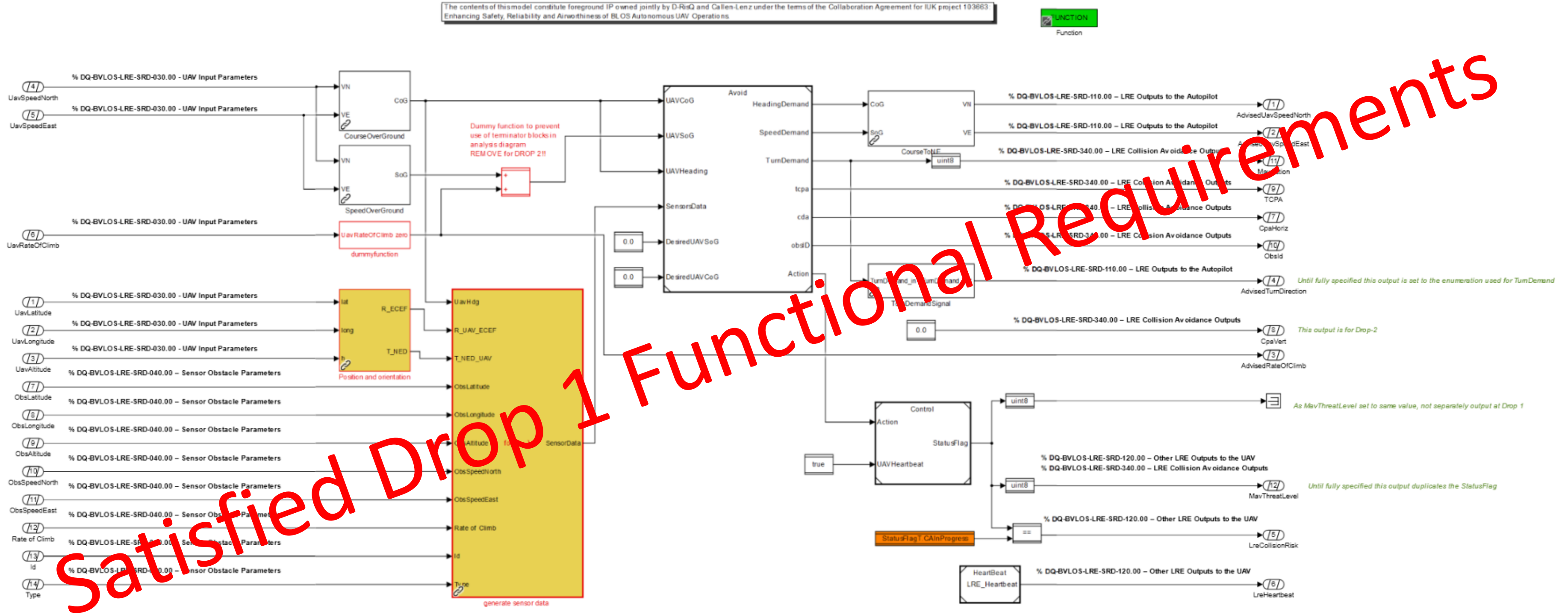
	Objective		Activity	Claim
	Description	**Ref**	**Ref**	
1	Low-level requirements comply with high-level requirements.	FM.6.3.a FM.6.3.2.a	FM.6.3.2	All except for the review of derived requirements
2	Low-level requirements are accurate and consistent.	FM.6.3.b FM.6.3.c FM.6.3.2.b	FM.6.3.2	Supports accuracy and consistency claims along with unambiguity
3	Low-level requirements are compatible with target computer.	FM.6.3.d FM.6.3.2.c	FM.6.3.2	Review items include resource use
4	Low-level requirements are verifiable.	FM.6.3.e FM.6.3.2.d	FM.6.3.2	Automatically provides formal semantics for verification
5	Low-level requirements conform to standards.	FM.6.3.f FM.6.3.2.e	FM.6.3.2	Encapsulates a design standard
6	Low-level requirements are traceable to high-level requirements.	FM.6.3.g FM.6.3.2.f	FM.6.3.2	Automates trace information to requirements expressed in Kapture
7	Algorithms are accurate.	FM.6.3.h FM.6.3.2.g	FM.6.3.2	Accuracy can be checked using Modelworks against requirements expressed in Kapture

Modelworks & Table FM.A-4 - Architecture

	Objective		Activity	Claim
	Description	**Ref**	**Ref**	
8	Software architecture is compatible with high-level requirements.	FM.6.3.3.a	FM.6.3.3	Checks that the architecture does not conflict with requirements expressed in Kapture.
9	Software architecture is consistent.	FM.6.3.c FM.6.3.3.b	FM.6.3.3	Checks control and data flow.
10	Software architecture is compatible with target computer.	FM.6.3.d FM.6.3.3.c	FM.6.3.3	Can check some aspects; remainder require review.
11	Software architecture is verifiable.	FM.6.3.e FM.6.3.3.d	FM.6.3.3	Automatically provides formal semantics
12	Software architecture conforms to standards.	FM.6.3f FM.6.3.3e	FM.6.3.3	Encapsulates a design standard
13	Software partitioning integrity is confirmed.	FM.6.3.3.f	FM.6.3.3	Modelworks can check partition integrity

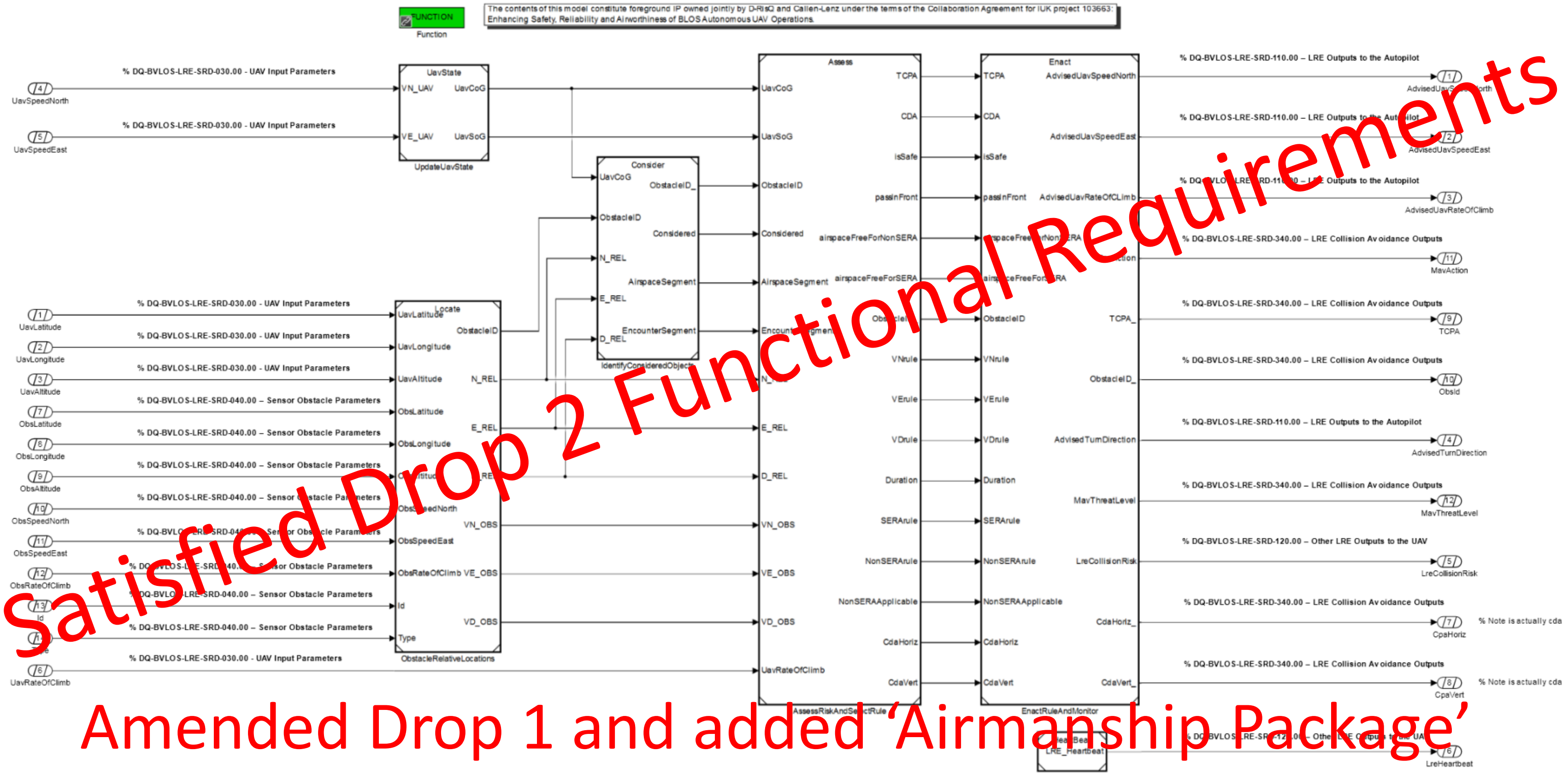
D-RisQ
SOFTWARE SYSTEMS

Drop 1 Simulink Model



D-RisQ SOFTWARE SYSTEMS

Drop 2 Simulink Model



Satisfied Drop 2 Functional Requirements

Amended Drop 1 and added ‘Airmanship Package’

RESULTS

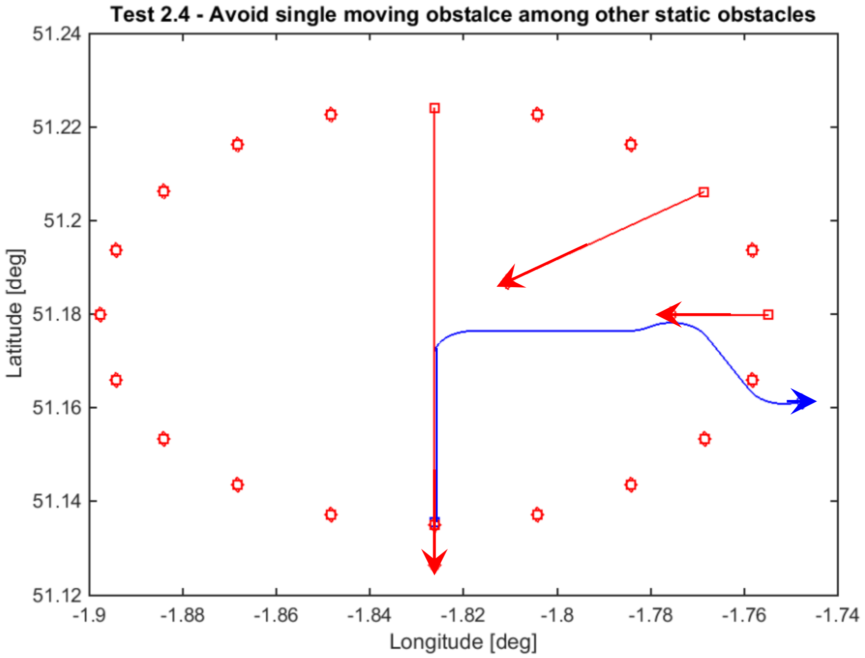
D-RisQ SOFTWARE SYSTEMS

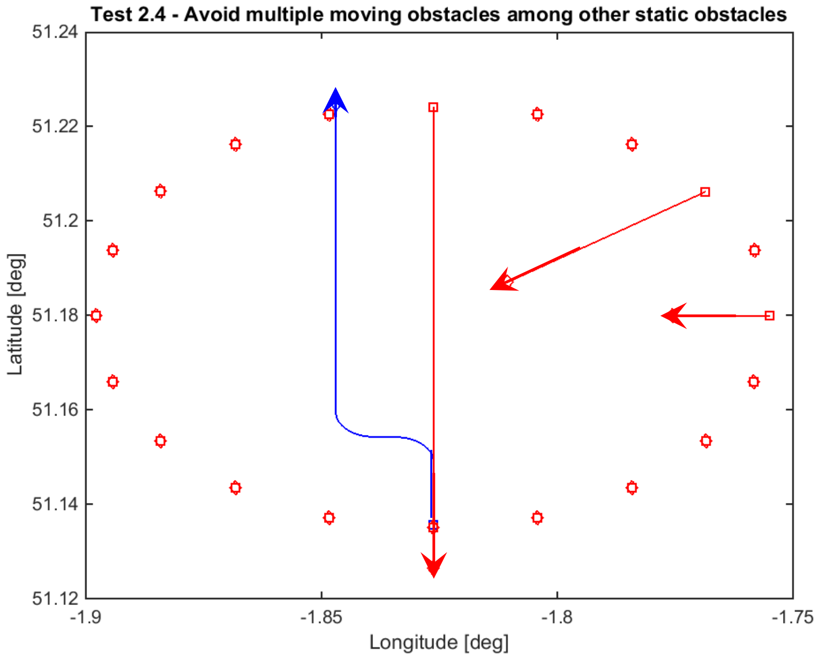
LRE Assumptions

- Any obstacle detected is assumed to be real
  - This is a sensor issue/sensor fusion issue
  - Might mean LRE reacts to false targets, but that’s safe
- Behaviour rules may not be the ‘right rules’
  - All we needed to show is that the LRE implements the rules
  - Adjustment to eg parameters can be easily made and incorporated

1st Flight Trial – 26 July 2018

Behavioural Changes Drop1: Drop2

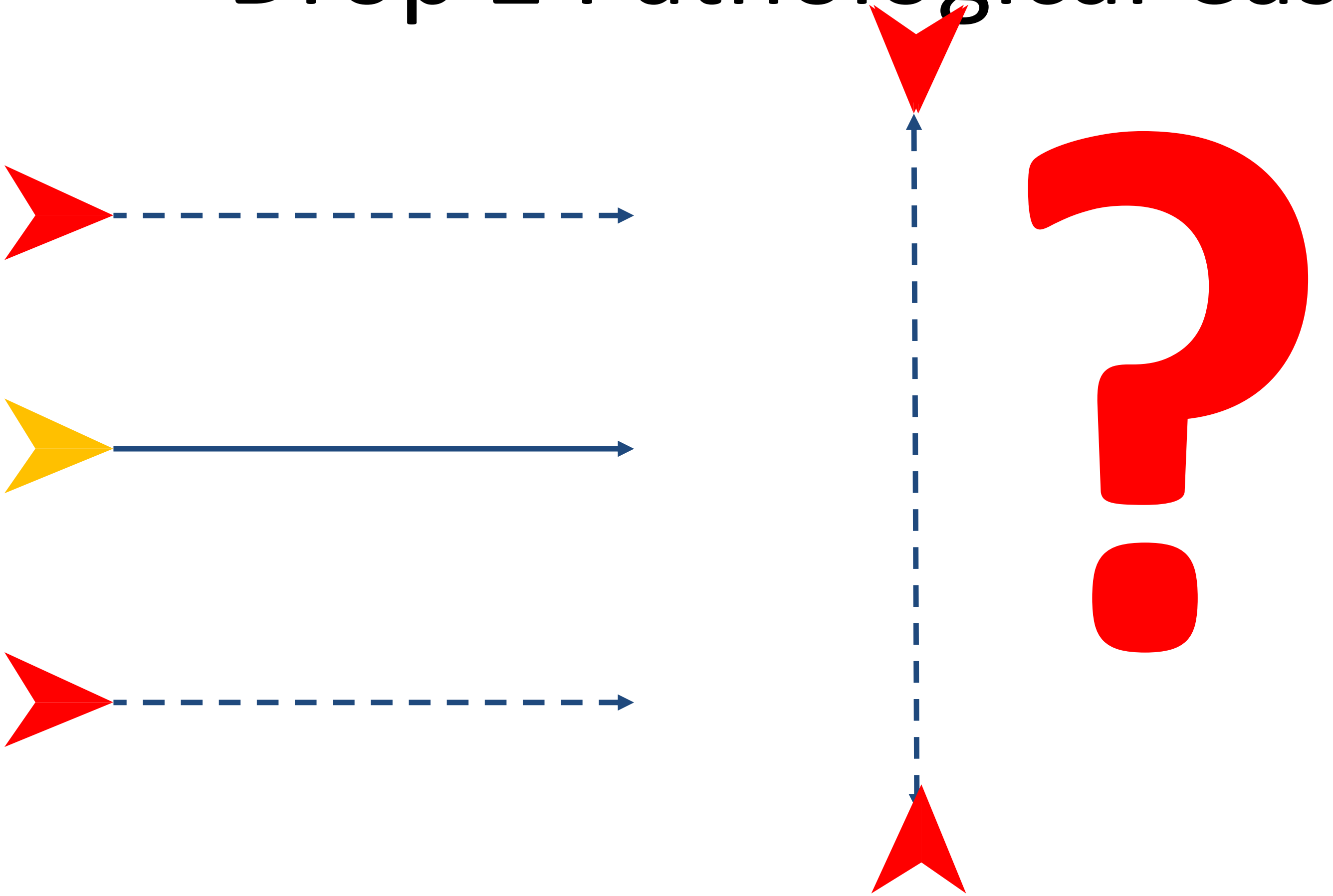




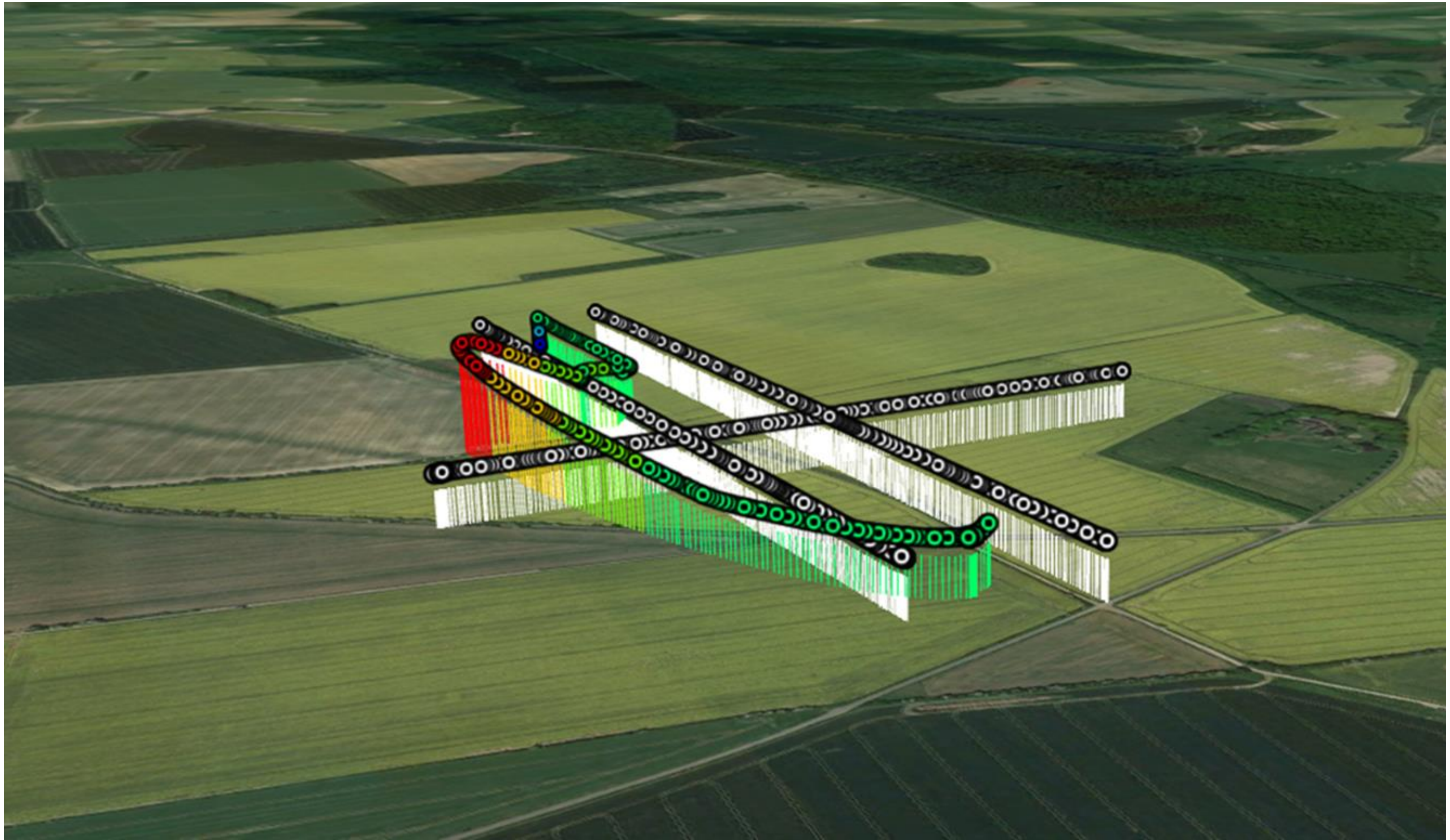
Drop 1 SERA: Starboard 90
+ Further manoeuvres

Drop 2(Permitted) non-SERA: Port 90
Most effective manoeuvre

Drop 2 Pathological Case

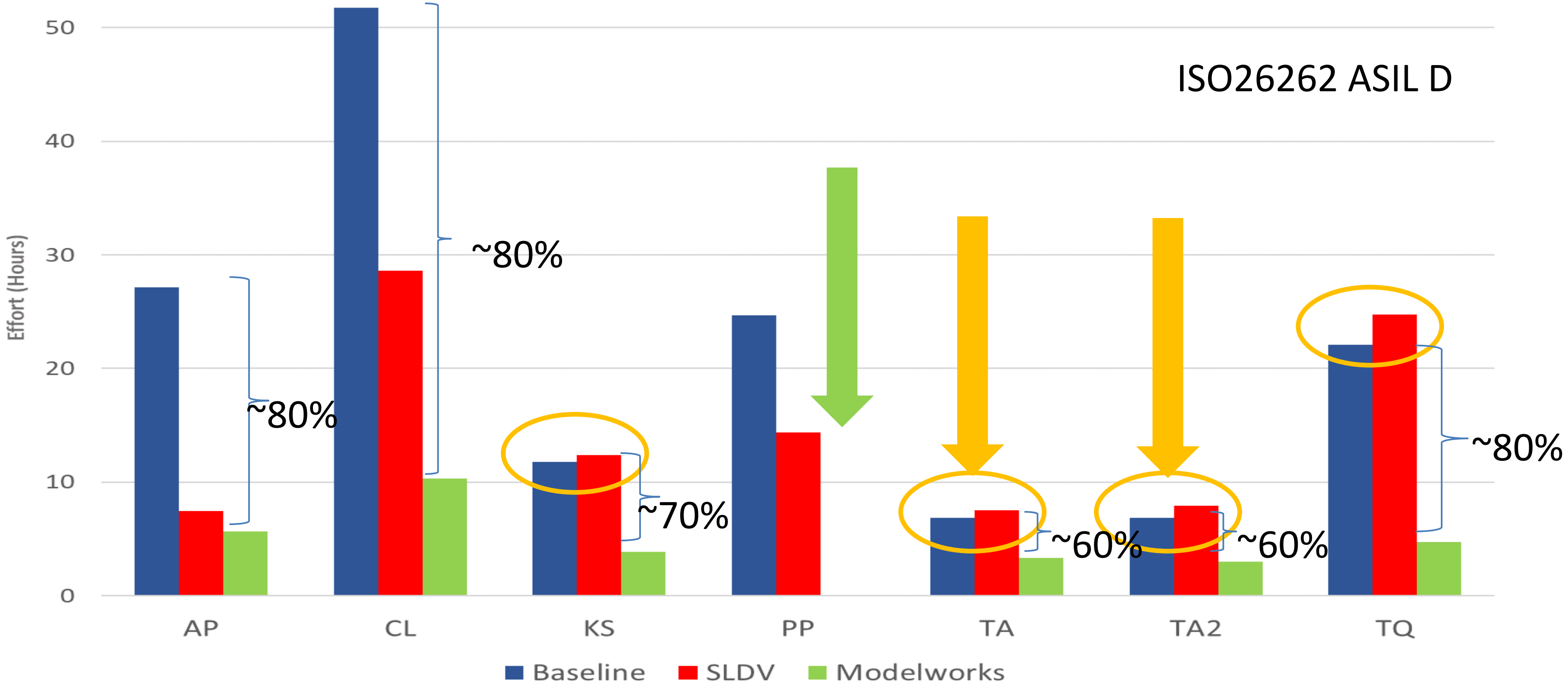


Simulation



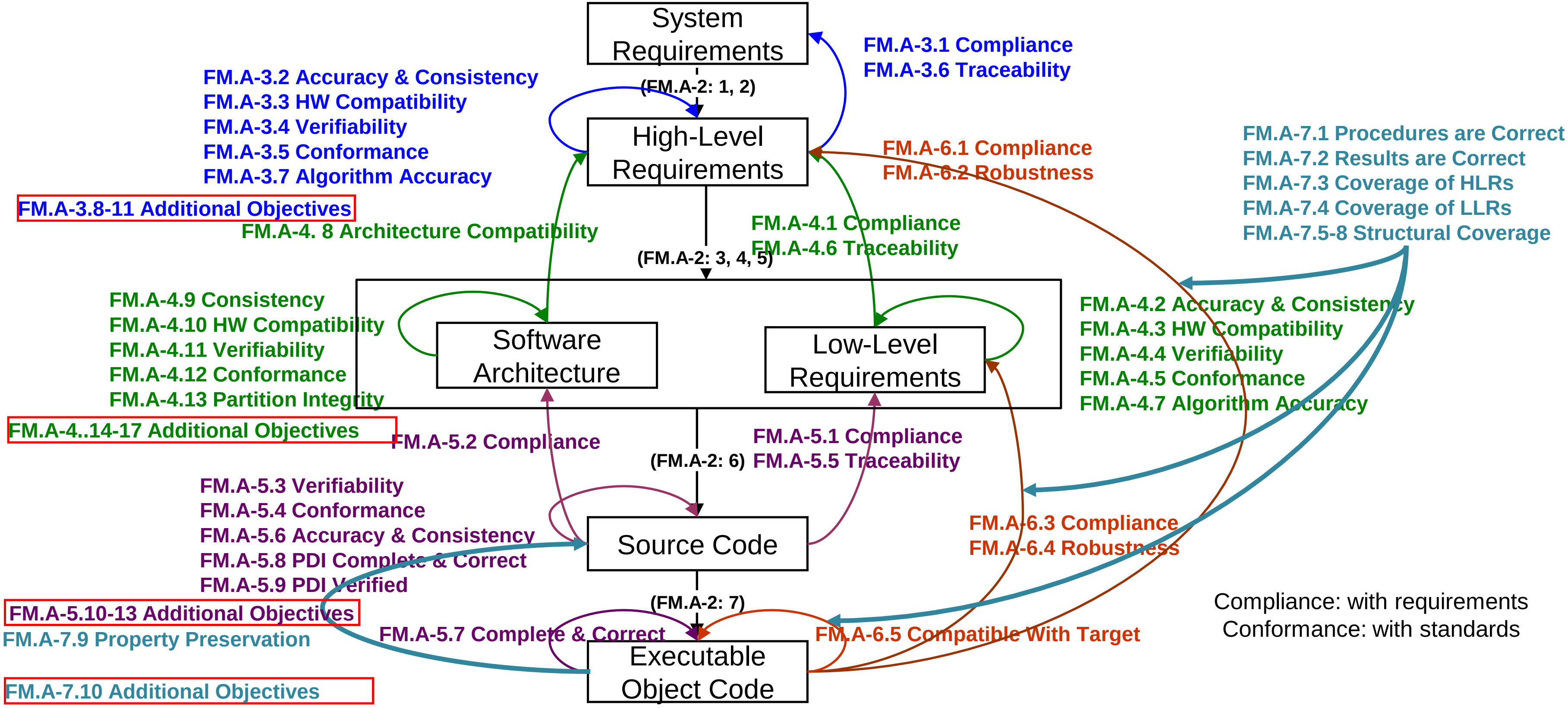
COST SAVINGS

Effort per Model – PICASSOS (2017)



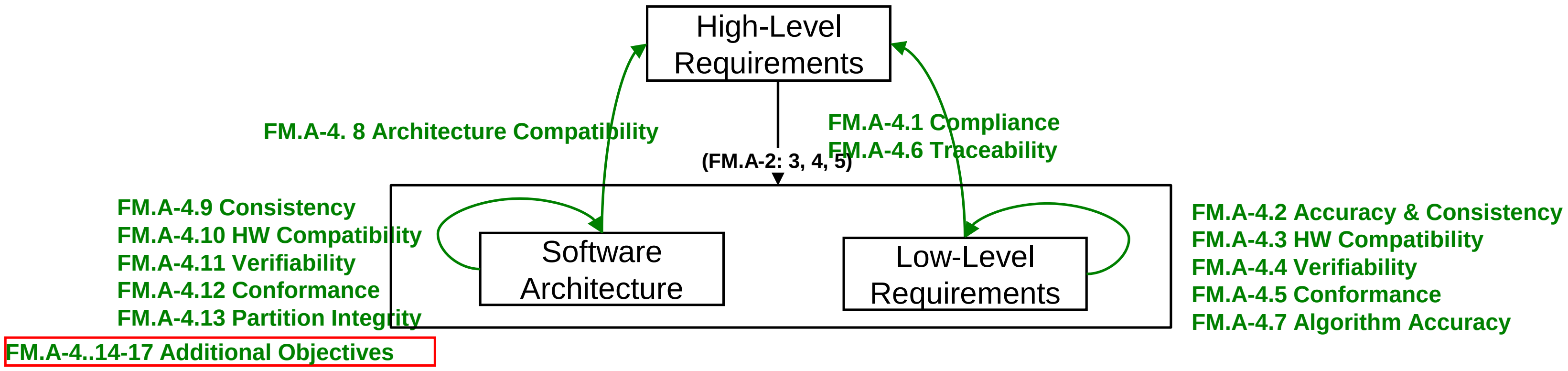
PICASSOS

Verification Objectives – DO-333

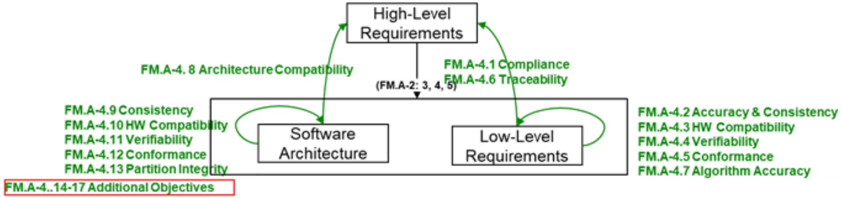


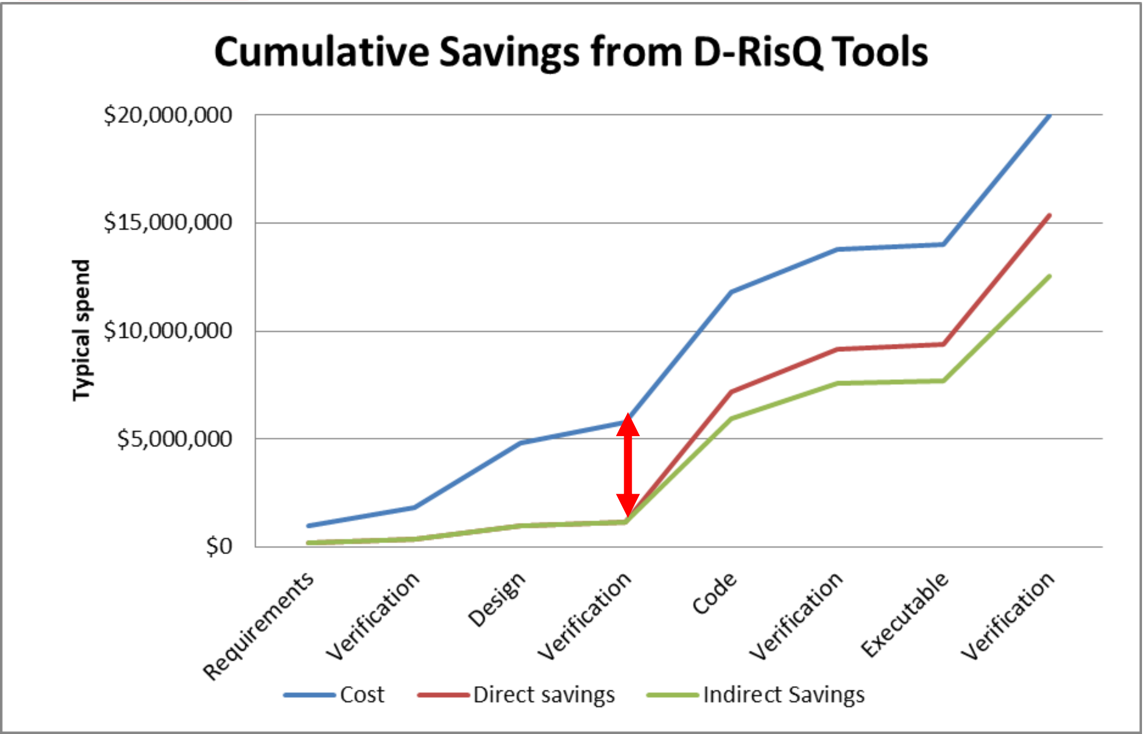
Compliance: with requirements
Conformance: with standards

Verification Objectives – DO-333



Verification Objectives – DO-333





Previous trials have shown that we can make circa 80% direct savings in this area

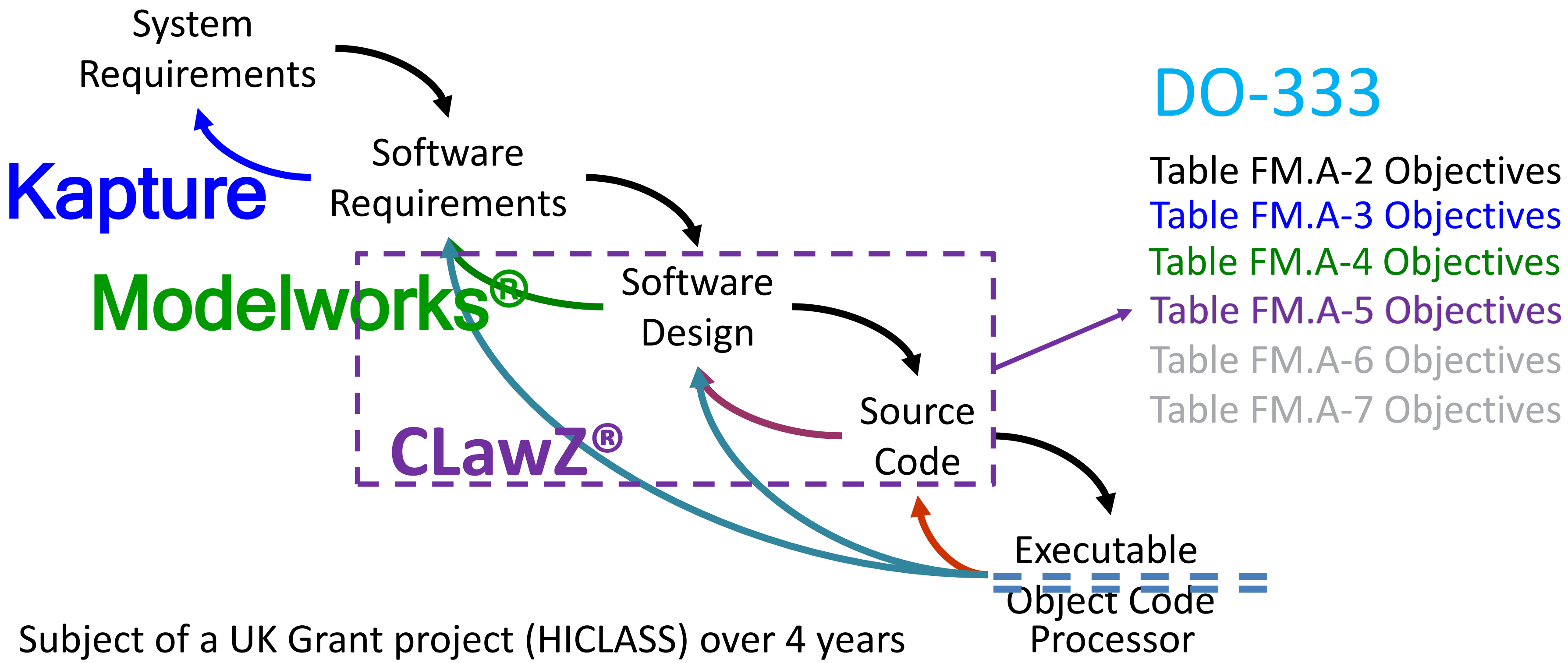
~25%

~33%

There are indirect savings to be had later in the life cycle

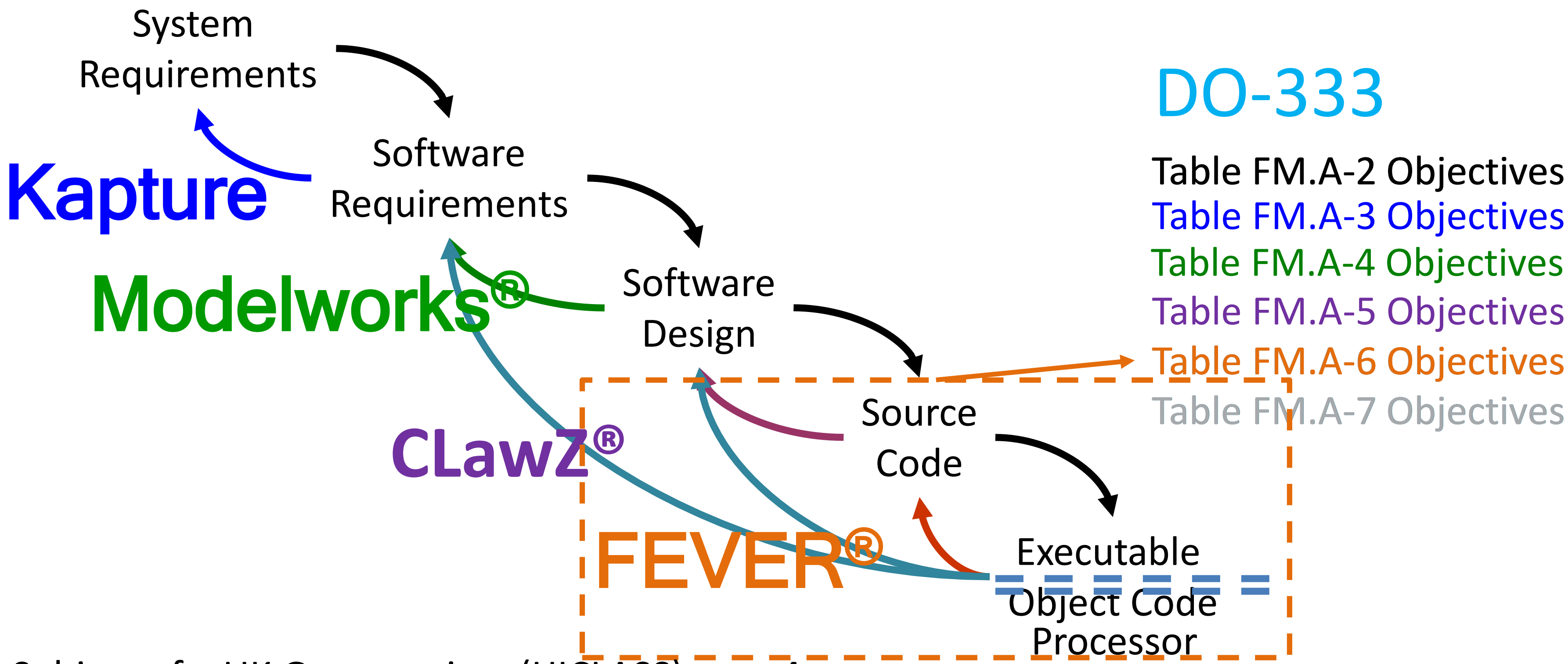
FUTURE PLANS

Systems, Software and Certification



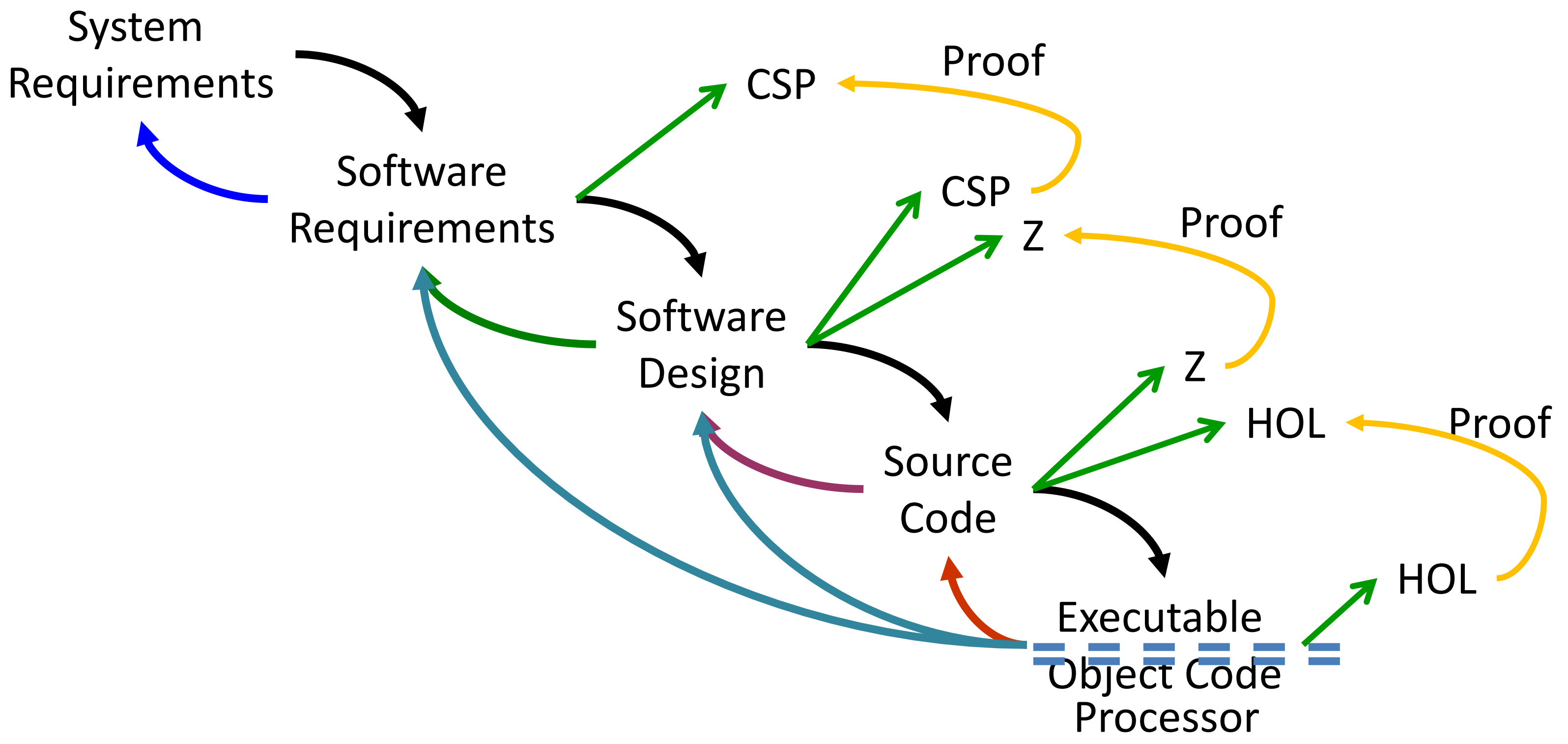
Subject of a UK Grant project (HICLASS) over 4 years

Systems, Software and Certification

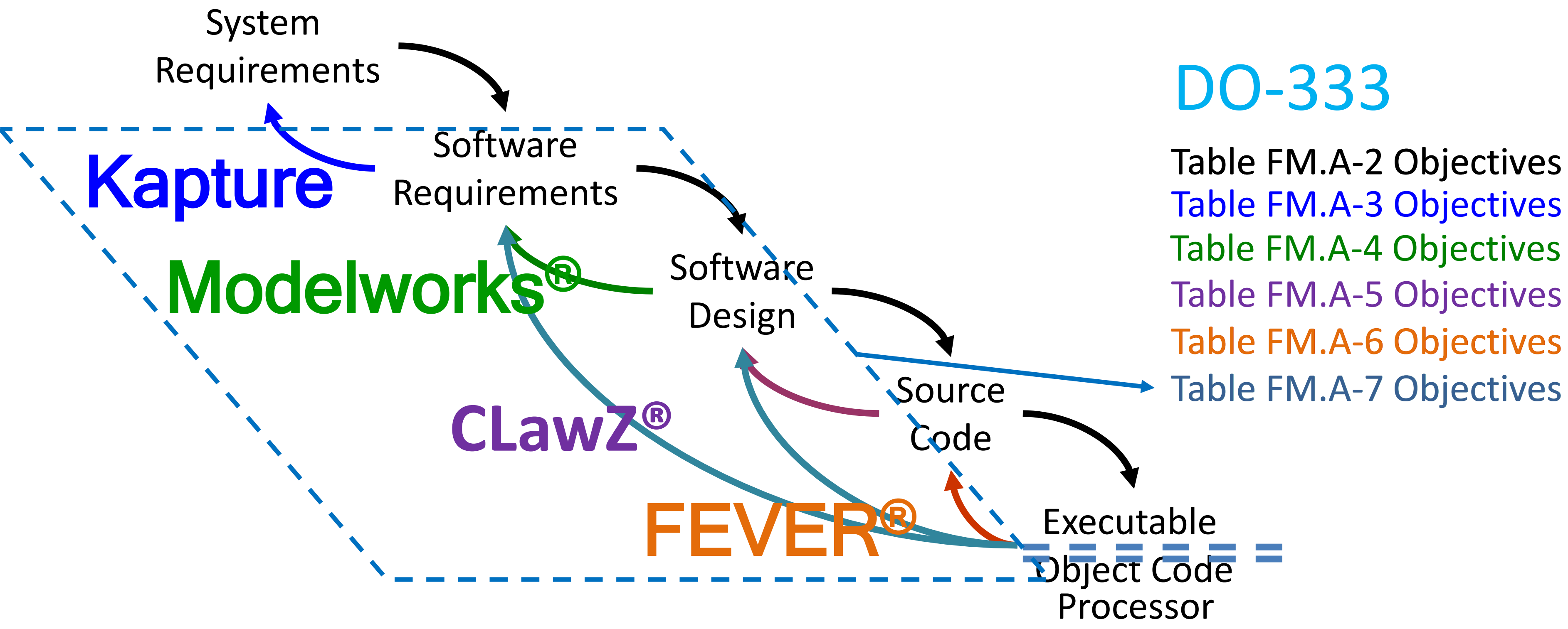


Subject of a UK Grant project (HICLASS) over 4 years

Systems, Software and Certification



Systems, Software and Certification



D-RisQ SOFTWARE SYSTEMS

D-RisQ & Table FM.A-7 - Coverage

	Objective		Activity	Claims
	Description	**Ref**	**Ref**	Use of complete D-RisQ toolset
FM 1	Formal analysis cases and procedures are correct.	FM.6.7.2.a FM.6.7.2.b	FM.6.7.2	Use of complete D-RisQ toolset
FM 2	Formal analysis results are correct and discrepancies explained.	FM.6.7.2.c	FM.6.7.2	Use of complete D-RisQ toolset
FM 3	Coverage of high-level requirements is achieved.	FM.6.7.1.a	FM.6.7.1.1	Use of complete D-RisQ toolset
FM 4	Coverage of low-level requirements is achieved.	FM.6.7.1.b	FM.6.7.1.1	Use of complete D-RisQ toolset
FM 5-8	Verification coverage of software structure is achieved.	FM.6.3 FM.6.3.4.e	FM.6.7.1.2 FM.6.7.1.3 FM.6.7.1.4 FM.6.7.1.5	Use of complete D-RisQ toolset in addition to an informal analysis (dead code)
FM9	Verification of property preservation between source and object code	FM.6.7.f	FM.6.7	FEVER provides proof of property preservation
FM10	Formal method is correctly defined, justified, and appropriate	FM.6.2.1	FM.6.2.1.a FM.6.2.1.b FM.6.2.1.c	Use of complete D-RisQ toolset

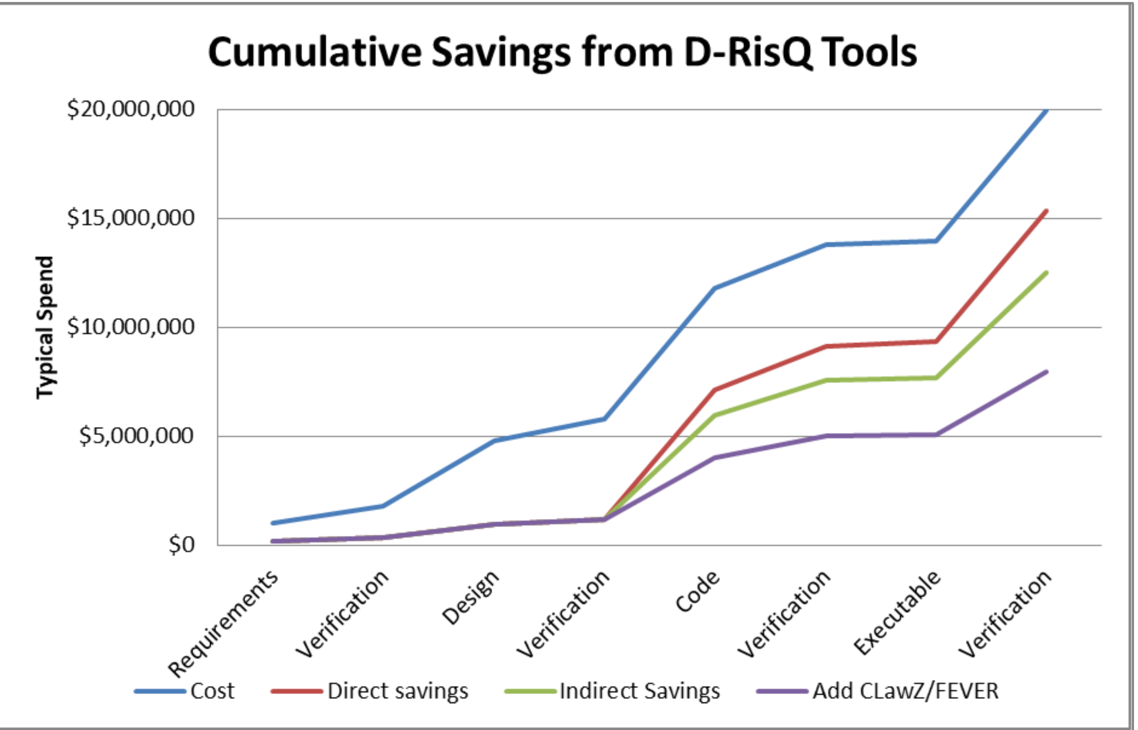
Verification Objectives – DO-333

Add automated code verification …

Table FM.A-5 Objectives

Table FM.A-6 Objectives

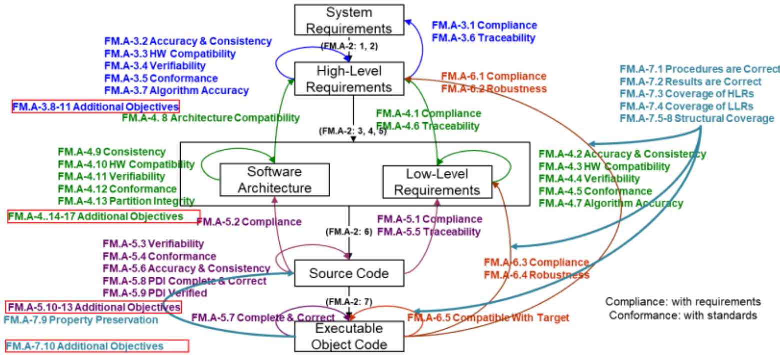
Table FM.A-7 Objectives



~25%

~33%

~60%



WRAP UP

Future Exploitation (Air)

- Build upon ESRA BVLOS project with Callen-Lenz
- Project to develop an assurance framework for swarms
  - Uses off-the-shelf swarm algorithm for demonstration
  - Included a formal verification of:
    - Overall swarm behaviour: normal, failures and collision avoidance
- Future civil and military applications

D-RisQ SOFTWARE SYSTEMS

Maritime

- Provision of advanced manoeuvring monitoring for underwater vehicles using BVLOS principles
  - Being tetherless is the crucial aspect
    - Requires ‘supervised’ autonomy
    - High integrity software
  - In 2 use cases:
    - Nuclear decommissioning
    - Off-shore
- Surface vessel developments also…on the horizon!

D-RisQ SOFTWARE SYSTEMS

Summary

- There are 3 things necessary to make the autonomous unmanned vehicle market:
  - Development of high integrity decision making software is necessary for autonomous UAVs (and other vehicles)
    - Whatever their size/task
  - Has to be at an affordable cost
  - Has to be to internationally recognised software standards
- Achieving all 3 will open up the market; less than all 3 will not
- D-RisQ products Kapture and Modelworks are already showing major benefits
- Future development of CLawZ and FEVER will result in further significant savings

D-RisQ SOFTWARE SYSTEMS

Thank you

D-RisQ

SOFTWARE SYSTEMS

Changing the way the world does software